Fossil SCM

Fossil Version Control
A Users Guide

Jim Schimpf

Document Number: PAN-20100424
Revision Number: 1.80
15 November 2016

Pandora Products.
215 Uschak Road
Derry, PA 15627

and Pandora Products

215 Uschak Road

Derry, PA 15627

Phone: 724.539.1276

Fax: 724.539.1276

Web: http://www.fossil-scm.org/schimpf-book/index

Email: jim.schimpf@gmail.com

Additional Contributors

- Marilyn Noz - Editor
- Wolfgang Stumvoll - Fossil Merge use and Windows use
- javalinBCD@gmail.com - found bugs
- arnel_legaspi@msn.com - found bugs

Document Revision History

Use the following table to track a history of this documents revisions. An entry should be made into this table for each version of the document.

Version	Author	Description	Date
0.1	js	Initial Version	24-Apr-2010
0.2	js	Finishing up Single User Chapter	27-Apr-2010
0.3	js	Working on introduction chapter	30-Apr-2010
0.4	js	Adding multiuser chapter	1-May-2010
0.5	mn	Adding editorial corrections [ebf40b842a]	4-May-2010
0.6	js	Adding Command sections [e11399d575]	8-May-2010
0.7	js	English & spelling corrections	19-May-2010
0.8	js	Spelling fixes	30-May-2010
0.9	ws	Using Fossil merge [db6c734300]	2-Jun-2010
1.0	js/ws	Put Fossil merge first in handling fork	3-Jun-2010
1.1	mn	Fixes in multiple user chapter [e8770d3172]	4-Jun-2010
1.2	js	Start advanced use chapter [2abc23dae5]	4-Jun-2010
1.3	mn	English corrections Chapter 1 [8b324dc900]	5-Jun-2010
1.4	mn	Sections 2.1 & 2.2 corrections [0b34cb6f04]	7-Jun-2010
1.5	js	Move close leaf to adv use [2abc23dae5]	7-Jun-2010
1.6	js	Convert Advanced chapter to forks and branching	13-Jun-2010
1.7	js/tr	Add note about IP port usage [a62efa8eba]	8-Jul-2010
1.71	javelin	Check on mispelling section 1.1 [637d974f62]	15-Sep-2011
1.72	anon	Fix absolute path in image regs [d54868853b]	15-Sep-2011
1.73	anon	Fix fossil create section 2.2.5 [36772d90a5]	15-Sep-2011
1.74	anon	Push/Pull described incorrectly [1b930fced6]	15-Sep-2011
1.75	arnel	Commands might be changed [4aaf1f78bb]	15-Sep-2011
1.76	js	Handle tickets [6477027434], [9274528931] and [837310e00a]	2-Nov-2013
1.77	js	Handle [0c9ebf37b7] with better explaination	3-Nov-2013
1.78	js	Chapter on Chiselapp [137b3f918c]	3-Nov-2013
1.79	js	Update on Chiselapp chapter	28-Dec-2014
1.80	js	Update to Fossil 1.36	15-Dec-2016

Contents

List of Figures

Foreward

This book was started in 2010 using a Dual processor Mac 550 MHz PowerPC running Mac OSX 10.5. Current revisions are now done on MacBook Pro (Intel) running MacOS 10.12.1 at 2.5GHz. It was originally done using Fossil version 1.27 and now revisions to the book are using Fossil 1.36. Because of this some of the pictures here will look slightly different in your copy of Fossil but the maintainers have been careful to update the esthetics but not the functions of the various pages. So you will see different, better looking pages but the functions will be the same.

1 Source Control & Why you need it

1.1 What is it

A source control system is software that manages the files in a project. A project (like this book or a software application) usually has a number of files. You then put all these files in a directory and at times subdivide even that with subdirectories. At any particular time this set of files in their present edited state make up a working version of the project. If other people are using the project you give them this and as is usually the case, they find problems and you fix them. This results in another version of the project slightly changed which goes through the cycle again. (This is why books have editions and software has versions...)

Software developers on large projects with multiple developers could see this cycle and realized they needed a tool to control the changes. With multiple developers sometimes the same file would be edited by two different people changing it in different ways or records of what got changed would be lost. It was hard to bring out a new release of the software and be sure that all the bugs were fixed and enhancements made.

A tool called Source Code Control System [6] was developed at Bell Labs in 1972 to track changes in files. It would remember each of the changes made to a file and store comments about why this was done. It also limited who could edit the file so conflicting edits would not be done. [5]

This was important but developers could see more was needed. They needed to be able to save the state of all the files in a project and give it a name (i.e., Release 3.14). As software projects mature you will have a released version of the software being used as well as bug reports written against it, while the next release of the software is being developed adding new features. The source control system would have to handle what are now called branches. One branch has say Version 1 which is released but continues to have fixes added to create Version 1.1, 1.2, etc. At the same time you also have another branch which contains Version 2 with new features added under construction.

In 1986 the open source Concurrent Version Control system CVS [3] was developed. This system could label groups of files and allow multiple branches (i.e. versions) simultaneously. There have been many other systems developed since them some open source and some proprietary.

Fossil which was originally released in 2006 [4] is a easy to install version control system that also includes a trouble ticketing system (Figure 17 on page 20), a wiki (Figure 2.6 on page 23) and self hosted web server (Figure 5 on page 11). It's home page is here `http://www.fossil-scm.org/`.

1.2 Why do it ?

Why do you want to use a source control system? To use one restricts your freedom, you won't be able to create files, delete files or, move files between directories at random. Making changes in your code becomes a check list of steps that must be followed carefully.

With all those hassles why do it? The biggest answer is freedom (huh ?). By following the procedures of a source control system you gain the freedom to modify your code any way you want. How does that follow ? One of the most horrible feelings as a developer is the "It worked yesterday" syndrome. That is, you had code that worked just fine and now it doesn't. You have a very helpless feeling of how do you get back to working code. With a source control system and careful adherence to procedures you can just go back in time and get yesterday's code. Then, starting from known good code you can figure out what happened.

Having a source control system also gives you the freedom to experiment, "let's try that radical new technique", and if it doesn't work it's easy to just go back to the previous state.

The rest of this book is a user manual for the Fossil version control system that does code management and much much more. It runs on multiple OS's and is FREE. It is simple to install as it has only one executable and the repositories it creates are a single file that is easy to back up and are usually only 50% the size of the original source.

1.2.1 How to get it

If this has interested you then you can get a copy of the Fossil executable here `http://www.fossil-scm.org/download.html`. There are Linux, Mac, and Windows executable links on this page. Also there is a source Tarball from which you can compile from source. This web site is self-hosted by Fossil itself (see Section 3 on page 27).

1.3 Source control description

This next section is useful if you have not used source control systems before. I will define some of the vocabulary and explain the basic ideas of source control.

1.3.1 Check out systems

When describing the grandaddy of source control systems, like SCCS I said it managed the changes for a single file and also prevented multiple people from working on the same file at the same time. This is representative of a whole class of source control systems. In these you have the idea of "checking-out" a file so you can edit it. At the same time while other people using the system can see who is working on the file they are prevented from touching it. They can get a read-only copy so they can say build software but only the "owner" can edit it. When done editing the "owner" checks it back in then anyone else could work on on it. At the same time the system has recorded who had it and the changes made to it.

This system works well in small groups with real time communication. A common problem is that a file is checked out by some one else and **you** have to make a change in it. In a small group setting, just a shout over the cube wall will solve the problem.

1.3.2 Merge systems

In systems represented by CVS or Subversion the barrier is not getting a file to work on but putting it back under version control. In these systems you pull the source code files to a working directory in your area. Then you edit these files making necessary changes. When done you commit or check them back into the repository. At this point they are back under version control and the system knows the changes from the last version to this version.

This gets around the problem mentioned above when others are blocked from working on a file. You now have the opposite problem in that more than one person can edit the same file and make changes. This is handled by the check-in process. There only one person at a time may check in a file. That being the case the system checks the file and if there are changes in repository file that are NOT in the one to be checked in stops the check in process. The system will ask if the user wants to merge these changes into his copy. Once that is done the new version of the file can be checked in.

This type of system is used on large projects like the Linux kernel or other systems where you have a large number of geographically distributed contributors.

1.3.3 Distributed systems

The representatives of two major systems we have described thus far are centralized. That is there is only one repository on a single server. When you get a copy of the files or check in files it all goes to just one place. These work and can support many, many users. A distributed system is an extension of this where it allows the repositories to be duplicated and has mechanisms to synchronize them.

With a single server users of the repository must be able to connect to it to get updates and for check ins of new code. If you are not connected to the server you are stuck and cannot continue working. Distributed systems allow you to have your own copy of the repository then continue working and when back in communication synchronize with the server. This is very useful where people take their work home and cannot access the company network. Each person can have a copy of the repository and continue working and re-sync upon return to the office.

1.3.4 Common Terms

The following is a list of terms I will use when talking about version control or Fossil.

Repository This is the store when the version controlled files are kept. It will be managed by a source control system

SCS Source control system, this is software that manages a group of files keeping track of changes and allowing multiple users to modify them in a controlled fashion

Commit In Fossil to store the current set of new and changed files into the repository.

Trunk The main line of code descent in a Fossil repository.

Branch A user defined split in the files served by an SCS. This allow multiple work points on the same repository. Older branches (versions) might have bug fixes applied and newer branches (versions) can have new features added.

Fork In Fossil an involuntary split in the code path, occurs when a file in the repository has changes not in a file to be committed.

2 Single User Fossil Use

2.1 Introduction

If you have read this far and are at least persuaded to try, you will want to put one of your projects under software control using Fossil. This chapter is set up to lead you through that task and show you how to adapt your development to using this tool. The assumption is made in this section that you will be the only person using the repository, you are the designer, developer, and maintainer of this project. After you are comfortable using the tool, the next section will show how you use it when you have multiple people working on a project. The rest of this book and most of the pictures are based on Fossil version 1.27 [13ad130920] 2013-09-11 11:43:49 UTC, Then Fossil version 1.36 [c24373934d] 2016-10-24 14:59:33 UTC (the latest as of this edit) was used for the Fossil Commands chapter.

2.2 Creating a repository

2.2.1 Introduction

In the spirit of "eating one's own dog food" we will use this book as the project we are going to manage with Fossil. The book is a directory of text files (we are writing it using LYX [2]) and my working area looks like this:

```
FOSSIL/

    This directory holds all my Fossil repositories

FossilBook/

    outline.txt    - Book design
    fossilbook.lyx - The book
    Layout
        fossil.png  - The Fossil logo (image on title page)
    Research
        fossilbib.bib - Working bibliography
        History
```

```
                    CVC-grune.pdf - Historical paper about CVS
                    RCS-A System for Version Control.webloc - RCS bookmark
                    SCCS-Slideshow.pdf - Historical paper about SCCS
                    VCSHistory -pysync ... .webloc - History of version control
```

This took just an hour or so to start preliminary research and build the framework. Since that's about all I'm going to do today I want to build a repository and put all this stuff under Fossil control.

2.2.2 Create Repository

I have a directory called FOSSIL in which I keep all my repositories, Fossil doesn't care but it helps me to keep them all in one place so I can back them up. First I need to create a new repository for the book. This is done using the command line after I move into the Fossil book directory.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil new ../FOSSIL/FossilBook.fossil
project-id: 2b0d35831c1a5b315d74c4fd8d532b100b822ad7
server-id:  0149388e5a3109a867332dd8439ac7b454f3f9dd
admin-user: jim (initial password is "ec3773")
```

Figure 1: Create Repository

I create my repositories with the extension .fossil, this will be useful later with the server command (See Figure 80 on page 70). When the create happened it assigned an initial password with an admin user of "jim" (i.e., me).

2.2.3 Connect Repository

The repository is created but is empty and has no connection to the book directory. The next step is to open the repository to the book directory with the **open** command.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil open ../FOSSIL/FossilBook.fossil
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/FossilBook.fossil
local-root:   /Users/jschimpf/Public/FossilBook/
server-code:  0149388e5a3109a867332dd8439ac7b454f3f9dd
checkout:     279dfecd3f0322f236a92a9a8f3c96acf327d8c1 2010-04-25 12:40:39 UTC
tags:         trunk
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra
Layout/fossil.png
Research/History/CVS-grune.pdf
Research/History/RCS--A System for Version Control.webloc
Research/History/SCCS-Slideshow.pdf
Research/History/VCSHistory - pysync - ....webloc
Research/fossilbib.bib
fossilbook.lyx
outline.txt
```

Figure 2: Open Repository & Check

The **open** command seemingly did nothing but checking with the **status** command shows the repository, the directory it's linked to and that we are hooked to the trunk of the store.

The **extra** command shows all the files in the directory that are NOT under control of Fossil. In this case that's all of them since we have not checked in anything.

2.2.4 Add and Initial Commit

I must now add all the relevant files into the repository with the **add** command. The Fossil add is recursive so if I add the top level files it will automatically recurse into the subdirectories like Layout and Research and get those files too. Before you do an add it pays to tidy up your directory so you don't accidentally add a bunch of transient files (like object files from a compile). It's easy to remove them later but a little tidying before hand can save you some work.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil add .
ADDED  Layout/fossil.png
ADDED  Research/History/CVS-grune.pdf
ADDED  Research/History/RCS--A System for Version Control.webloc
ADDED  Research/History/SCCS-Slideshow.pdf
ADDED  Research/History/VCSHistory - pysync ....webloc
ADDED  Research/fossilbib.bib
fossil: cannot add _FOSSIL_
ADDED  fossilbook.lyx
ADDED  outline.txt
```

Figure 3: Initial file add

I simply told fossil to do an add of the current directory (.) so it got all those files and all the files in the subdirectory. Note the _FOSSIL_ that it didn't add. This is the tag file that fossil keeps in a directory so it knows what repository it belongs to. Fossil won't add this file since it manages it, but everything else is fair game.

One final thing before I can quit for the day, these files have been added or rather they will be added to the repository when I commit it. That must be done and then we can relax and let Fossil manage things.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "Initial Commit"
New_Version: 8fa070818679e1744374bc5302a621490276d739
```

Figure 4: Initial Commit

I added a comment on the commit and it's a good idea to always do this. When later we see the timeline of the commits you will have notes to tell you what was done.

2.2.5 Fossil start up summary

- **fossil new <name>** Creates a new fossil repository

- **fossil open <repository>** While in a source directory connects this directory to the fossil repository
- **fossil add .** Will add (recursively) all the files in the current directory and all subdirectories to the repository
- **fossil commit -m "Initial Commit"** Will put all the currently added files into the repository.

2.3 Set Up User interface

One of the surprising features of Fossil is the web server. This allows it to have a GUI type user interface with no operating system specific code, the GUI is the web browser supplied by your OS. In the previous steps I checked my project in to a Fossil repository, next I have to prep the web interface for production use.

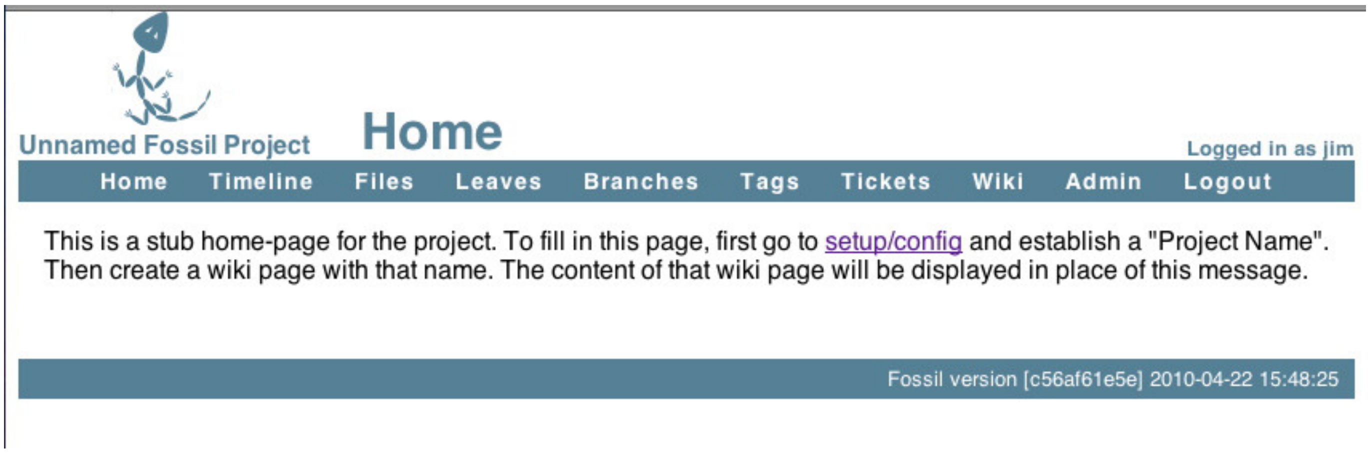
NOTE The Fossil web server uses port 8080 instead of the standard port 80 for all HTTP access. When run it will automatically start your Web browser and open the repository home page. Unfortunately my book work is done on a machine that already has Apache running on port 8080 so I will be using port 8081. I will always have to add an extra parameter on the UI command line to do this.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil ui -port 8081
```

Figure 5: Starting Webserver

NOTE: Newer versions of Fossil automatically find an open port and will give a message on the command line telling you what port number is used. You can still use the -port option if you want to control the port #.

This shows how it's started, as you can see I have picked port 8081 since I am locked out of port 8080. When I do this my browser starts and I am presented with the following home page.

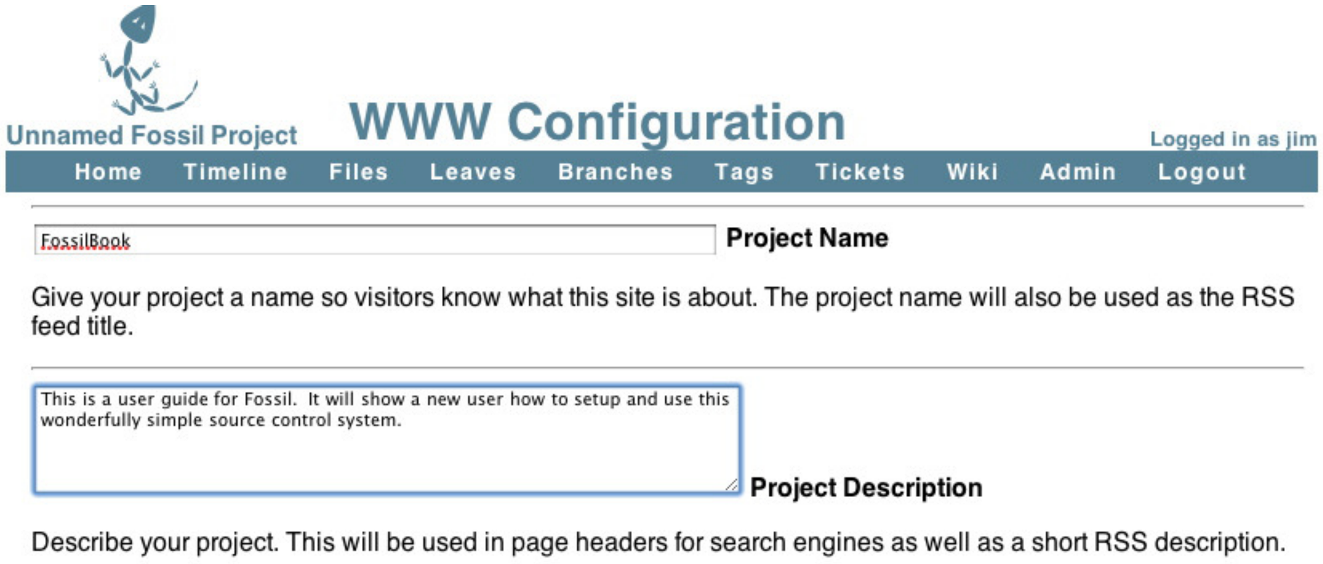

Unnamed Fossil Project
Home
Logged in as jim
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout

This is a stub home-page for the project. To fill in this page, first go to setup/config and establish a "Project Name". Then create a wiki page with that name. The content of that wiki page will be displayed in place of this message.

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 6: Initial Webserver page

Following the advice on the page I go to **setup/config**. I am going to do the minimum setup that you should do on all projects. As you get familiar with Fossil you will probably have many more things that you will customize for your taste but what follows are the only things you HAVE to do.


Unnamed Fossil Project
WWW Configuration
Logged in as jim
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout

FossilBook **Project Name**

Give your project a name so visitors know what this site is about. The project name will also be used as the RSS feed title.

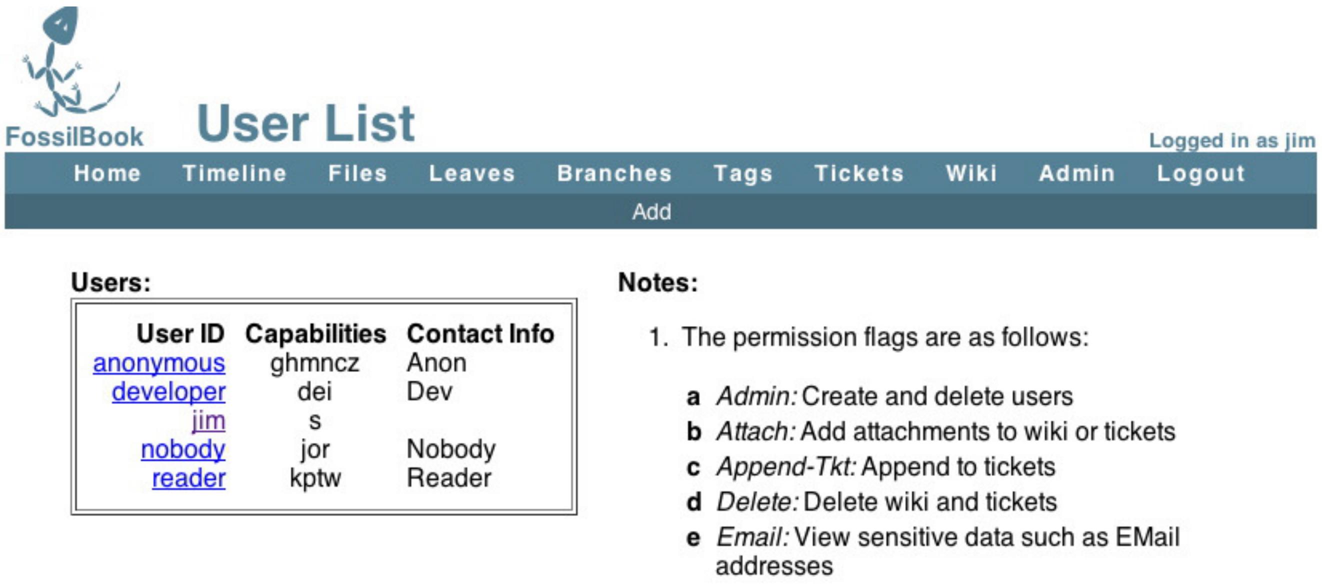
This is a user guide for Fossil. It will show a new user how to setup and use this wonderfully simple source control system. **Project Description**

Describe your project. This will be used in page headers for search engines as well as a short RSS description.

Figure 7: Initial Configuration

I have entered in a project name and put in a description, the project name will be the name of the initial Wiki page (see 2.6 on page 23) and the description is useful for others to see what you are doing here. Then I go to the bottom of the page and pick the **Apply Changes** button.

Next I pick the Admin tab (you can see it in the header bar) and the pick Users from that page. I am presented with the users and will use this to set the password of the project.



Users:

User ID	Capabilities	Contact Info
anonymous	ghmncz	Anon
developer	dei	Dev
jim	s	
nobody	jor	Nobody
reader	kptw	Reader

Notes:

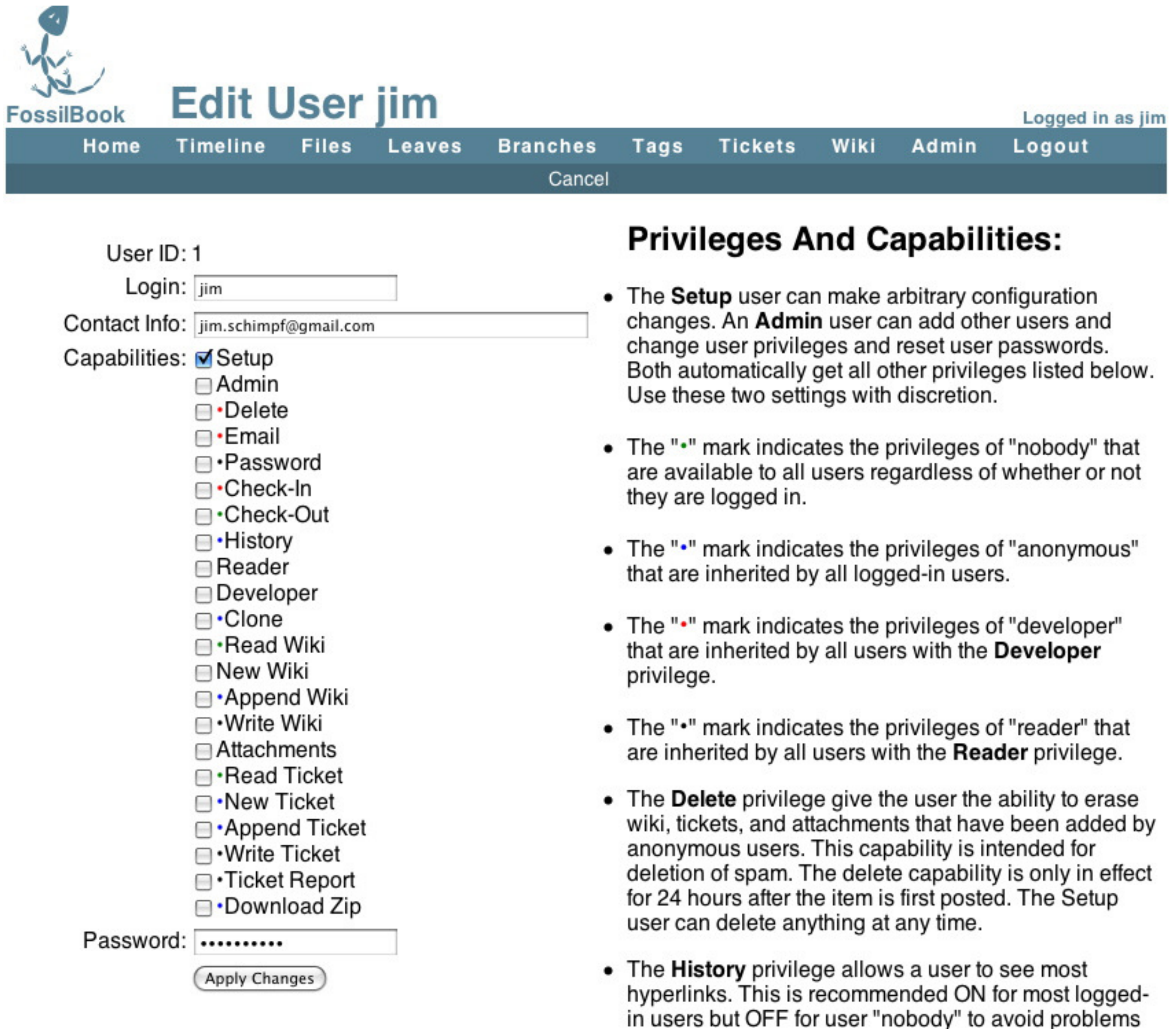
1. The permission flags are as follows:

   a *Admin:* Create and delete users
   b *Attach:* Add attachments to wiki or tickets
   c *Append-Tkt:* Append to tickets
   d *Delete:* Delete wiki and tickets
   e *Email:* View sensitive data such as EMail addresses

Figure 8: User Configuration

As you can see Fossil automatically configures a number of users beyond just the creator. The anonymous user you have already seen if you went to the Fossil web site to download the code. This user can view and get code but cannot commit code. On the right side of the page are the many options you can give to a user, it's worth reading it all when you set up your repository. The important one is me (jim) which has "s" or Super User Capabilities. This means I can do anything with the repository.

I will now edit the user Jim to make sure it has the settings I want. In this case you MUST set the password. Remember way back where Fossil set it during the create (Figure 1 on page 9), it's a very good idea to change this to something you can remember rather than the original random one.



Edit User jim

User ID: 1
Login: jim
Contact Info: jim.schimpf@gmail.com
Capabilities:
- ☑ Setup
- ☐ Admin
- ☐ •Delete
- ☐ •Email
- ☐ •Password
- ☐ •Check-In
- ☐ •Check-Out
- ☐ •History
- ☐ Reader
- ☐ Developer
- ☐ •Clone
- ☐ •Read Wiki
- ☐ New Wiki
- ☐ •Append Wiki
- ☐ •Write Wiki
- ☐ Attachments
- ☐ •Read Ticket
- ☐ •New Ticket
- ☐ •Append Ticket
- ☐ •Write Ticket
- ☐ •Ticket Report
- ☐ •Download Zip

Password: ••••••••••

Privileges And Capabilities:

- The **Setup** user can make arbitrary configuration changes. An **Admin** user can add other users and change user privileges and reset user passwords. Both automatically get all other privileges listed below. Use these two settings with discretion.
- The "•" mark indicates the privileges of "nobody" that are available to all users regardless of whether or not they are logged in.
- The "•" mark indicates the privileges of "anonymous" that are inherited by all logged-in users.
- The "•" mark indicates the privileges of "developer" that are inherited by all users with the **Developer** privilege.
- The "•" mark indicates the privileges of "reader" that are inherited by all users with the **Reader** privilege.
- The **Delete** privilege give the user the ability to erase wiki, tickets, and attachments that have been added by anonymous users. This capability is intended for deletion of spam. The delete capability is only in effect for 24 hours after the item is first posted. The Setup user can delete anything at any time.
- The **History** privilege allows a user to see most hyperlinks. This is recommended ON for most logged-in users but OFF for user "nobody" to avoid problems

Figure 9: Super User Setup

I have put in my contact information (e-mail address) and while you cannot see it I have typed in a password that I will remember. Then I applied the changes.

Now the repository is ready for further work, it's rather bare bones at this point but the most important things are set up.

2.3.1 User interface summary

- **fossil ui** run in the source directory will start a browser based user interface to fossil.

- **fossil ui -port <IP port #>** Can be used if port 8080 if already in use on your system.
- On the first run it is important to configure your project with a name and set the password for yourself.

2.4 Update Repository

After writing the above section of the book I now have created a bunch of new files and changed some of the existing files in the repository. Before quitting for the day I should add these new files into the repository and commit the changes saving this milestone in the project.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra
#fossilbook.lyx#
Images/Single_user/config_initial.epsf
Images/Single_user/initial_page.epsf
Images/Single_user/jim_setup.epsf
Images/Single_user/user_config.epsf
fossilbook.lyx~
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/FossilBook.fossil
local-root:   /Users/jschimpf/Public/FossilBook/
server-code:  0149388e5a3109a867332dd8439ac7b454f3f9dd
checkout:     8fa070818679e1744374bc5302a621490276d739 2010-04-25 13:09:02 UTC
parent:       279dfecd3f0322f236a92a9a8f3c96acf327d8c1 2010-04-25 12:40:39 UTC
tags:         trunk
EDITED     fossilbook.lyx
```

Figure 10: Project Status

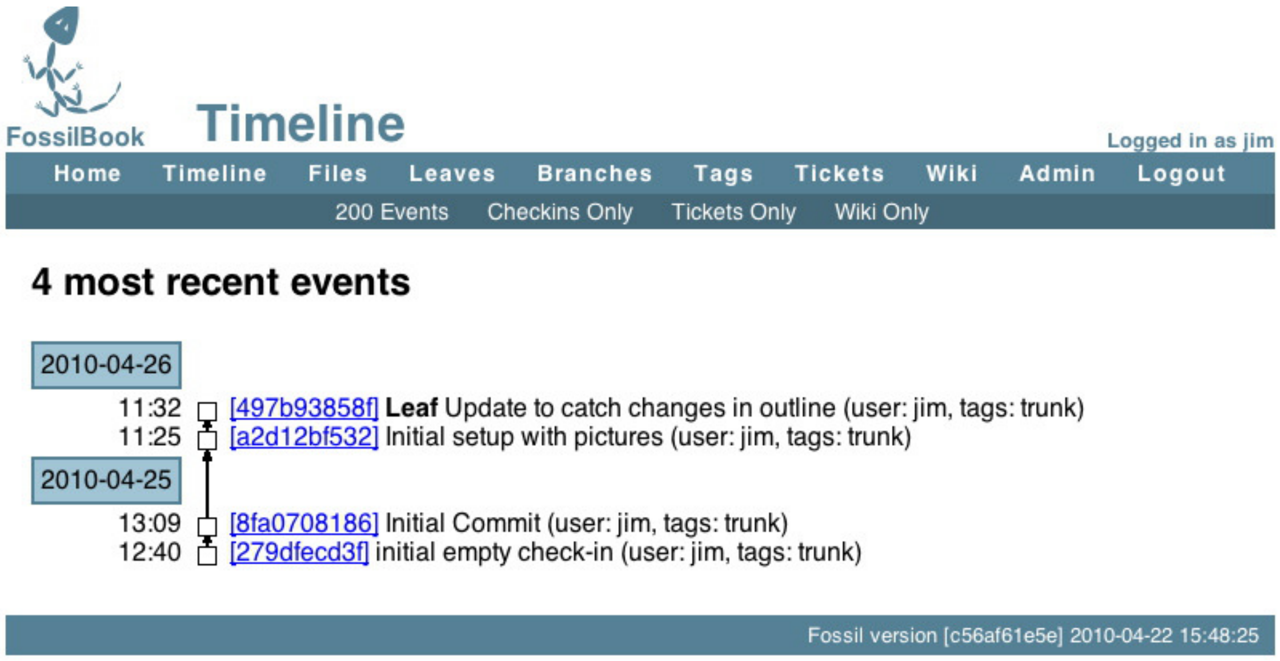
I run **fossil extra** to see these new files. The image files (those in Images/Single_user) I want to add, the other two files, #fossilbook.lyx# and fossilbook.lyx~, I don't want to add since they are temporary artifacts of LYX. I also ran **fossil status**. This shows changes to files that are already in the repository. There the only file changed is the book text itself, **fossilbook.lyx**.

All I have to do now is add in the directory Images which will add in the image files I want in the repository. Then I commit the changes to the repository and we can move on to other tasks of the day.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil add Images
ADDED  Images/Single_user/config_initial.epsf
ADDED  Images/Single_user/initial_page.epsf
ADDED  Images/Single_user/jim_setup.epsf
ADDED  Images/Single_user/user_config.epsf
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "Initial setup with pictures"
New_Version: a2d12bf532a089ee53578e3e17c6e732c0442f49
```

Figure 11: Update for new files

After doing this commit I can bring up the Fossil ui (see Figure 5 on page 11) and view the project Timeline by picking that tab on the Fossil header. We get this:



4 most recent events

2010-04-26

11:32 [497b93858f] **Leaf** Update to catch changes in outline (user: jim, tags: trunk)
11:25 [a2d12bf532] Initial setup with pictures (user: jim, tags: trunk)

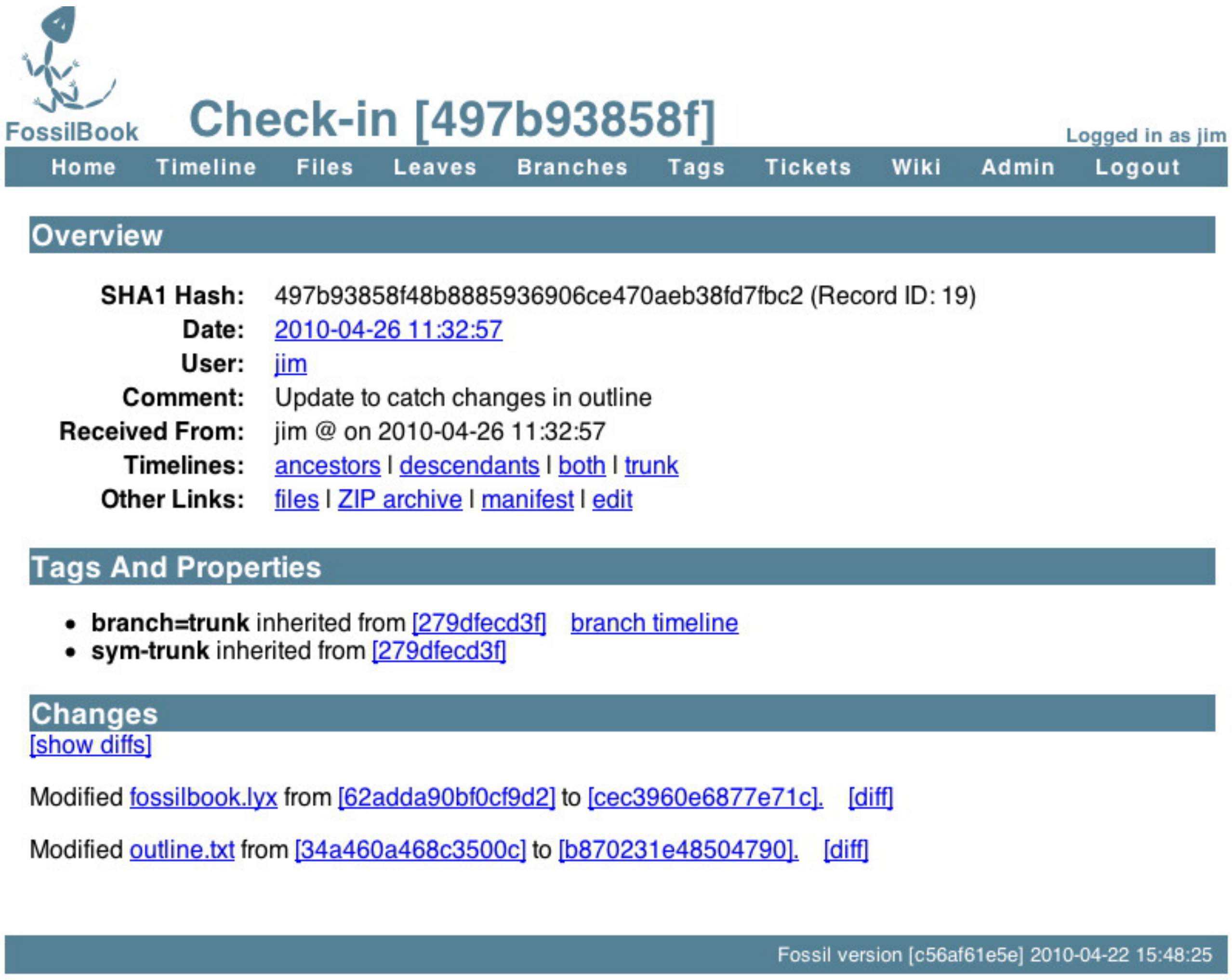
2010-04-25

13:09 [8fa0708186] Initial Commit (user: jim, tags: trunk)
12:40 [279dfecd3f] initial empty check-in (user: jim, tags: trunk)

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 12: Timeline

You can see all my check-ins thus far and you can see after the check-in from Figure 11 on the preceding page I did another check-in because I missed some changes in the outline. The check-ins are labeled with the first 10 digits of their hash value and these are active links which you can click to view in detail what was changed in that version.



Check-in [497b93858f]

Overview

SHA1 Hash:	497b93858f48b8885936906ce470aeb38fd7fbc2 (Record ID: 19)
Date:	2010-04-26 11:32:57
User:	jim
Comment:	Update to catch changes in outline
Received From:	jim @ on 2010-04-26 11:32:57
Timelines:	ancestors | descendants | both | trunk
Other Links:	files | ZIP archive | manifest | edit

Tags And Properties

- **branch=trunk** inherited from [279dfecd3f] branch timeline
- **sym-trunk** inherited from [279dfecd3f]

Changes
[show diffs]

Modified fossilbook.lyx from [62adda90bf0cf9d2] to [cec3960e6877e71c]. [diff]

Modified outline.txt from [34a460a468c3500c] to [b870231e48504790]. [diff]

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 13: Timeline Detail

I clicked on the very last check-in (the **LEAF**) and the display is shown above. There are many things you can do at this point. From the list of changed files you can pick the diff link and it will show in text form the changes made in that particular file. The "Other Links" section has a very useful ZIP Archive. Clicking this will download a ZIP of this version to your browser. You will find this useful if you want to get a particular version, in fact this is normally how you get a new version of Fossil from the `http://www.fossil-scm.org/`. The edit link will be used later to modify a leaf.

2.4.1 Update Summary

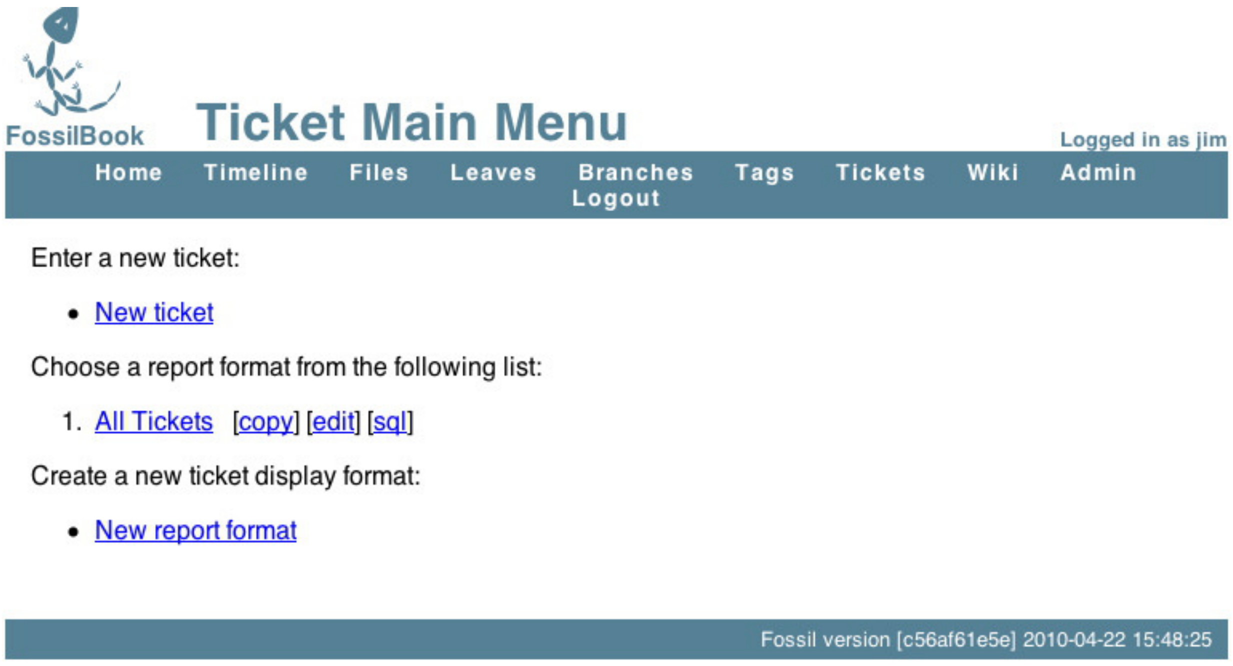
- **fossil status** and **fossil extra** will tell you the updated files and files not in the repository before you commit.

- **fossil commit - m "Commit comment"** Commits a change (or changes). It is very important to have a descriptive comment on your commit.

2.5 Tickets

Besides managing your code Fossil has a trouble ticket system. This means you can create a ticket for a problem or feature you are going to add to your system then track your progress. Also you can tie the tickets to specific check-ins of your files. For software this is very useful for bug fixes and feature additions. For example you can look for a bug in the ticket list then have it take you to the change that fixed the problem. Then you know exactly what you did and not have to be confused by other changes you might have made.

When you pick Tickets it will bring up this window. You can create a new ticket, look at the list, or generate a new report. Keeping things simple I will just use the All Tickets list for now.



Enter a new ticket:

- New ticket

Choose a report format from the following list:

1. All Tickets [copy] [edit] [sql]

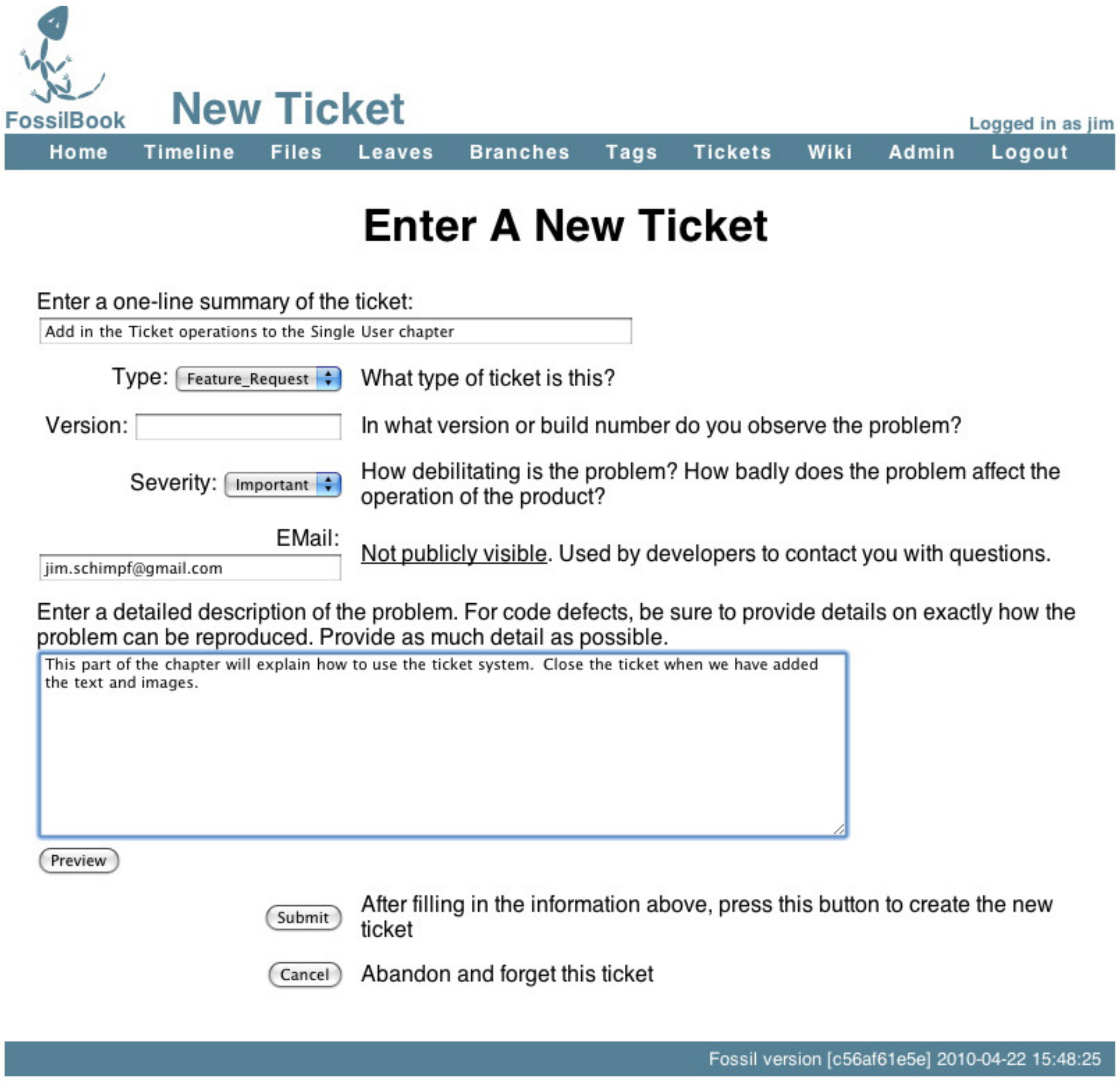
Create a new ticket display format:

- New report format

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 14: Initial Ticket Window

Picking "New Ticket" I get a form that I fill out like so:



Enter A New Ticket

Enter a one-line summary of the ticket:
Add in the Ticket operations to the Single User chapter

Type: Feature_Request What type of ticket is this?

Version: In what version or build number do you observe the problem?

Severity: Important How debilitating is the problem? How badly does the problem affect the operation of the product?

EMail: jim.schimpf@gmail.com Not publicly visible. Used by developers to contact you with questions.

Enter a detailed description of the problem. For code defects, be sure to provide details on exactly how the problem can be reproduced. Provide as much detail as possible.
This part of the chapter will explain how to use the ticket system. Close the ticket when we have added the text and images.

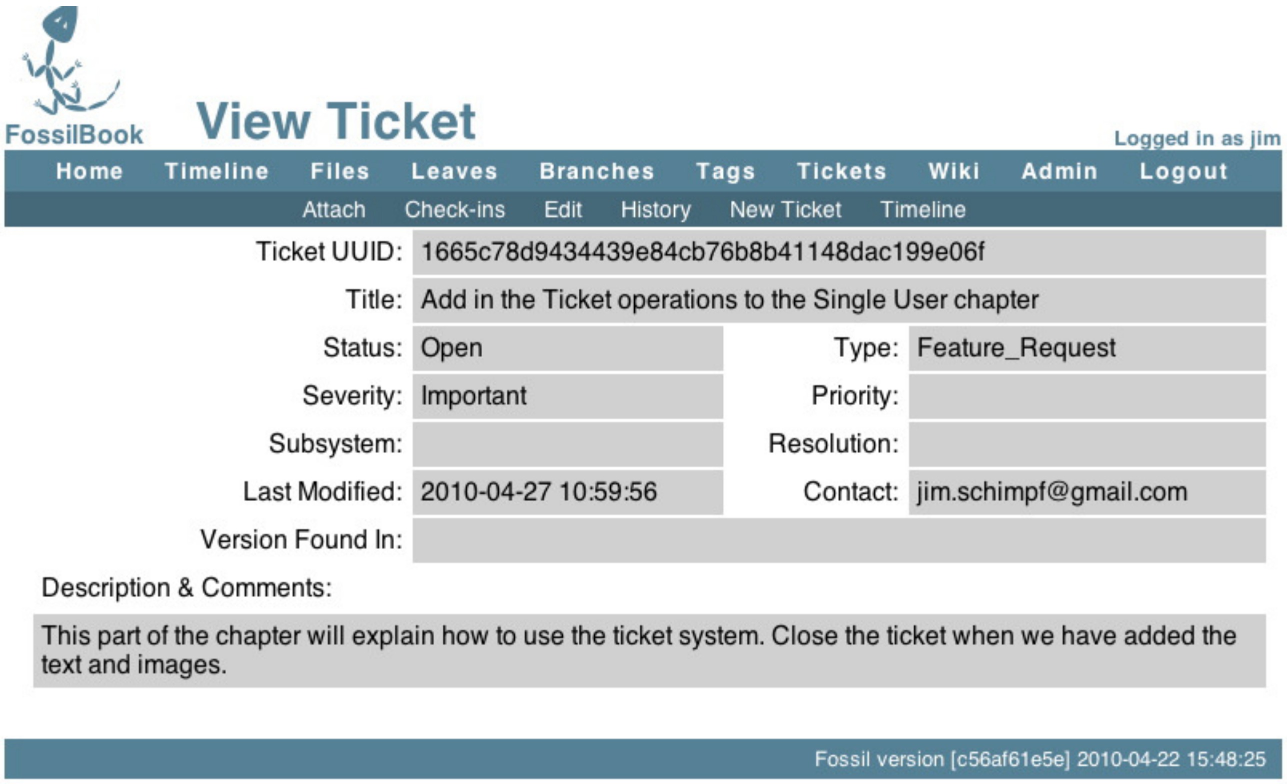
After filling in the information above, press this button to create the new ticket

Abandon and forget this ticket

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 15: Ticket Form

Pretty simple actually. You can put as much or as little detail in here as you wish, but remember this stuff might be really vital 6 weeks or 6 months from now so think of what you would want to know then. When I press Submit I get this showing what I entered.


FossilBook
View Ticket
Logged in as jim
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout
Attach Check-ins Edit History New Ticket Timeline

Ticket UUID:	1665c78d9434439e84cb76b8b41148dac199e06f		
Title:	Add in the Ticket operations to the Single User chapter		
Status:	Open	Type:	Feature_Request
Severity:	Important	Priority:	
Subsystem:		Resolution:	
Last Modified:	2010-04-27 10:59:56	Contact:	jim.schimpf@gmail.com
Version Found In:			

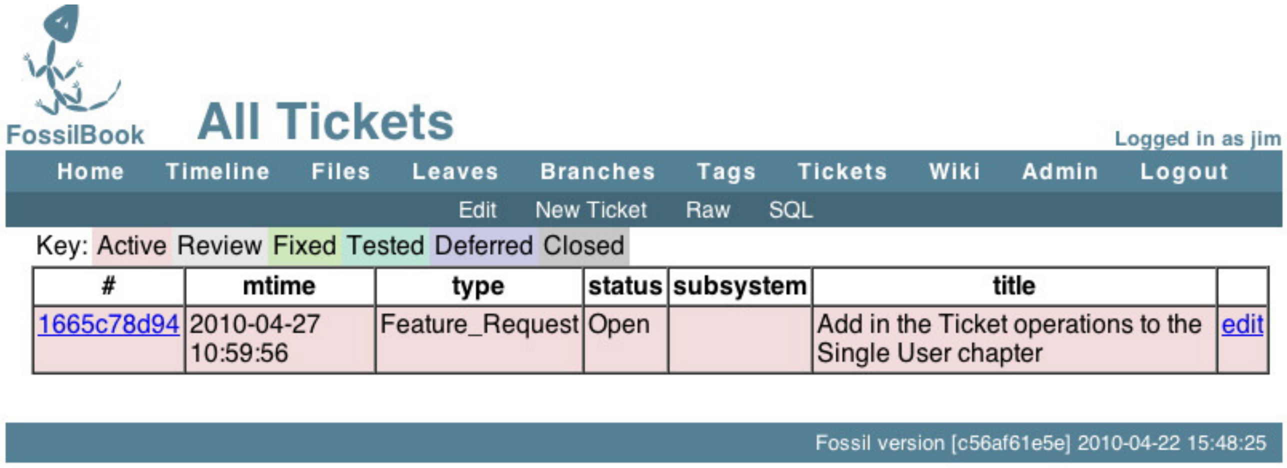
Description & Comments:

This part of the chapter will explain how to use the ticket system. Close the ticket when we have added the text and images.

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 16: Viewing a Ticket

Finally picking Tickets then “All Tickets” I can see my new ticket in the list marked as Open and in a distinctive color.


FossilBook
All Tickets
Logged in as jim
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout
Edit New Ticket Raw SQL

Key: Active Review Fixed Tested Deferred Closed

#	mtime	type	status	subsystem	title	
1665c78d94	2010-04-27 10:59:56	Feature_Request	Open		Add in the Ticket operations to the Single User chapter	edit

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 17: Ticket List with open ticket

I try, in handling tickets, to have links from ticket to the commit that addressed the problem and a link from the commit back to the offending ticket. This way looking at the ticket I can get to the changes made and from the timeline I can get the the ticket and its resolution. To do this I will make sure and put the 10 digit hash label from the ticket into the check-in comment and put a link in the resolved ticket to the check-in.

Since I have now written the chapter and put in all these images of what to do I can now add in all the new images to the repository and check this in as another completed operation. And I do that like this:

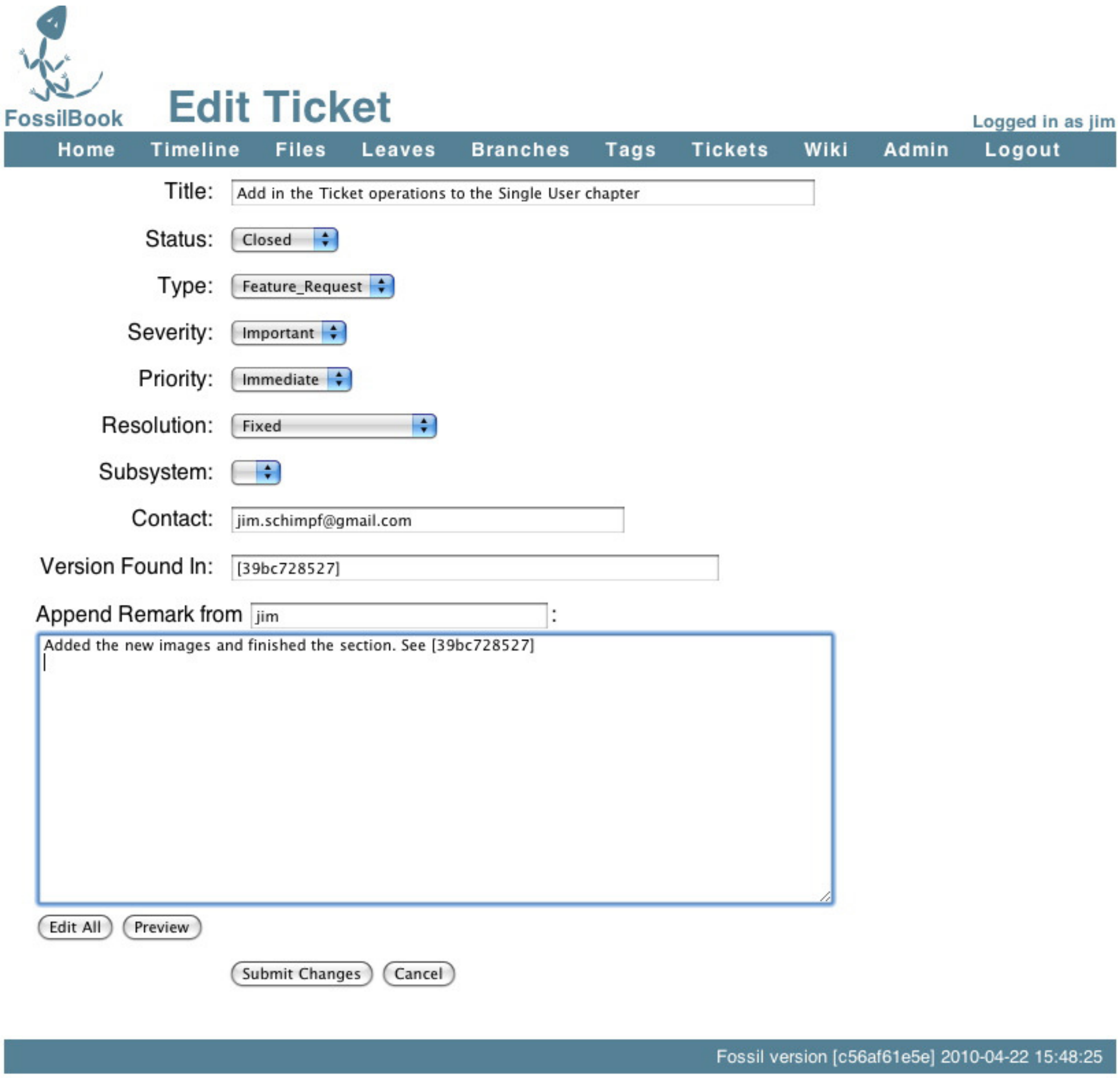
```
Pandora-2:jschimpf/Public/FossilBook] jim% fossil add Images/Single_user
ADDED  Images/Single_user/config_initial.epsf
ADDED  Images/Single_user/initial_page.epsf
ADDED  Images/Single_user/jim_setup.epsf
ADDED  Images/Single_user/ticket_form.epsf
ADDED  Images/Single_user/ticket_initial.epsf
ADDED  Images/Single_user/ticket_list.epsf
ADDED  Images/Single_user/ticket_submit.epsf
ADDED  Images/Single_user/timeline.epsf
ADDED  Images/Single_user/timeline_detail.epsf
ADDED  Images/Single_user/user_config.epsf
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "[1665c78d94] Ticket Use"
```

Figure 18: Ticket resolving check-in

First I added in all the new image files. I am lazy and just told it to add in all the files in the Single_user directory. I have previously added some of those like **config_initial.epsf** but Fossil is smart and knows this and won't add that one twice. Even though it shows it ADDED, it really didn't.

The commit line is very important, as you can see I put 10 digit hash code for the ticket in brackets in the comment. As we will see in the Wiki section this is a link to the Ticket, so when viewing the comment in the Timeline or elsewhere you can click the bracketed item and you would go to the ticket.

Now that I have the items checked in I have to close the ticket. I do that by clicking on its link in the ticket list, that will go the the View Ticket window as shown in Figure 17 on the previous page. From there I pick edit and fill it in as shown:



Title: Add in the Ticket operations to the Single User chapter

Status: Closed

Type: Feature_Request

Severity: Important

Priority: Immediate

Resolution: Fixed

Subsystem:

Contact: jim.schimpf@gmail.com

Version Found In: [39bc728527]

Append Remark from jim :

Added the new images and finished the section. See [39bc728527]

Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 19: Ticket Finish

I mark it as “Closed”. If you have code you can mark this as fixed, tested, or a number of other choices. Another very important step, I brought up the Timeline and copied the link for the commit I had just done in Figure 18 on the preceding page. By doing this my ticket is now cross linked with the commit and the commit has a link back to the ticket.

2.5.1 Ticket Summary

- Tickets are a useful way of reminding you what needs done or bugs fixed
- When you commit a change that affects a ticket put the 10 digit hash label of the ticket into the commit comment surrounded by brackets, e.g. [<10 digit hash>] so you can link to the ticket
- When you close the ticket put in the hash label of the commit that fixed it.

2.6 Wiki Use

As we saw in Figure 5 on page 11 Fossil has a browser based user interface. In addition to the pages that are built in you can add pages to web site via a wiki. This allows you to add code descriptions, user manuals, or other documentation. Fossil keeps all that stuff in one place under version control. A wiki is a web site where you can add pages and links from within your browser. You are given an initial page then if you type [newpage] this text will turn into a link and if clicked will take you to a new blank page. Remember in Figure 6 on page 12 that is the initial page for your project and from there you can add new pages. These pages are automatically managed by the Fossil version control system; you don't have to add or commit.

Since I did the setup on repository (see Figure 7 on page 12) the home page has changed to this:

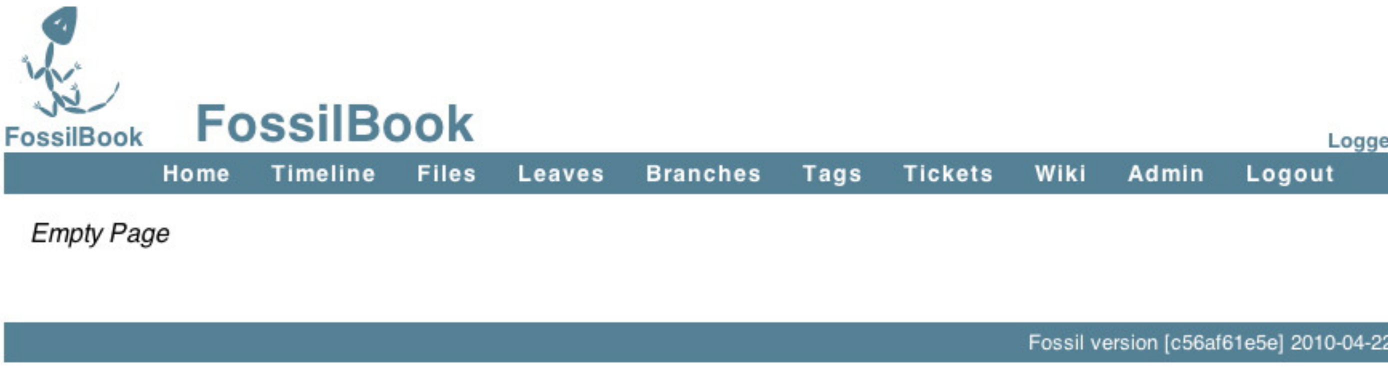
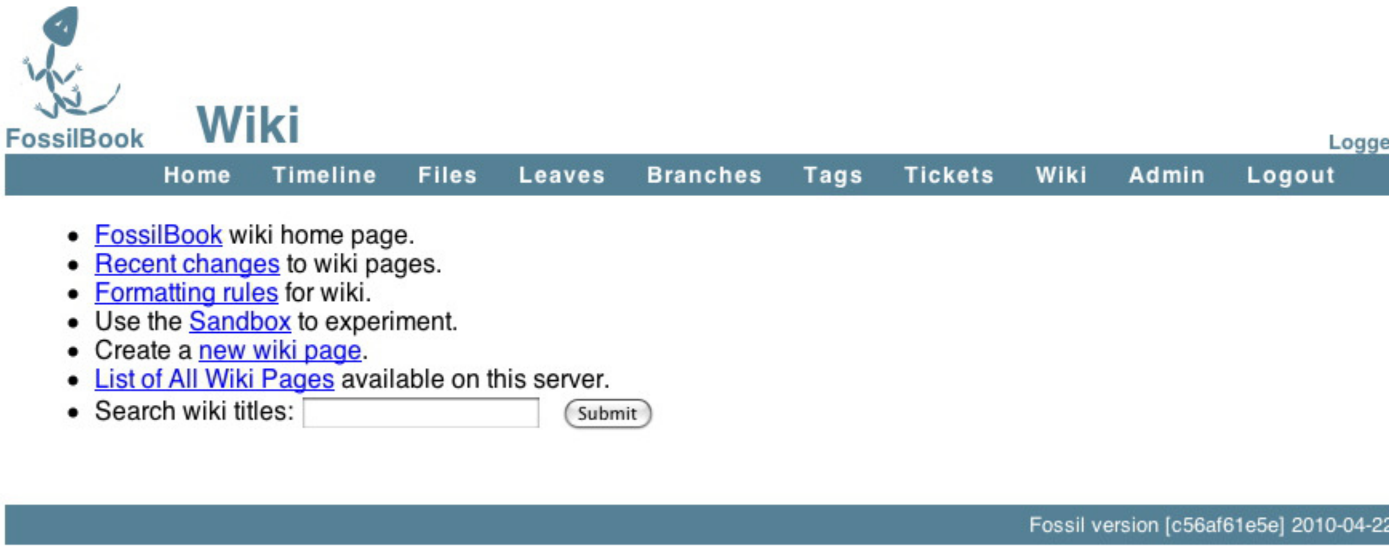


Figure 20: Blank Home Page

Not very helpful so the in rest of this chapter I will use the Wiki functions of Fossil to make this more useful. If I pick the Wiki item from the menu bar I get:



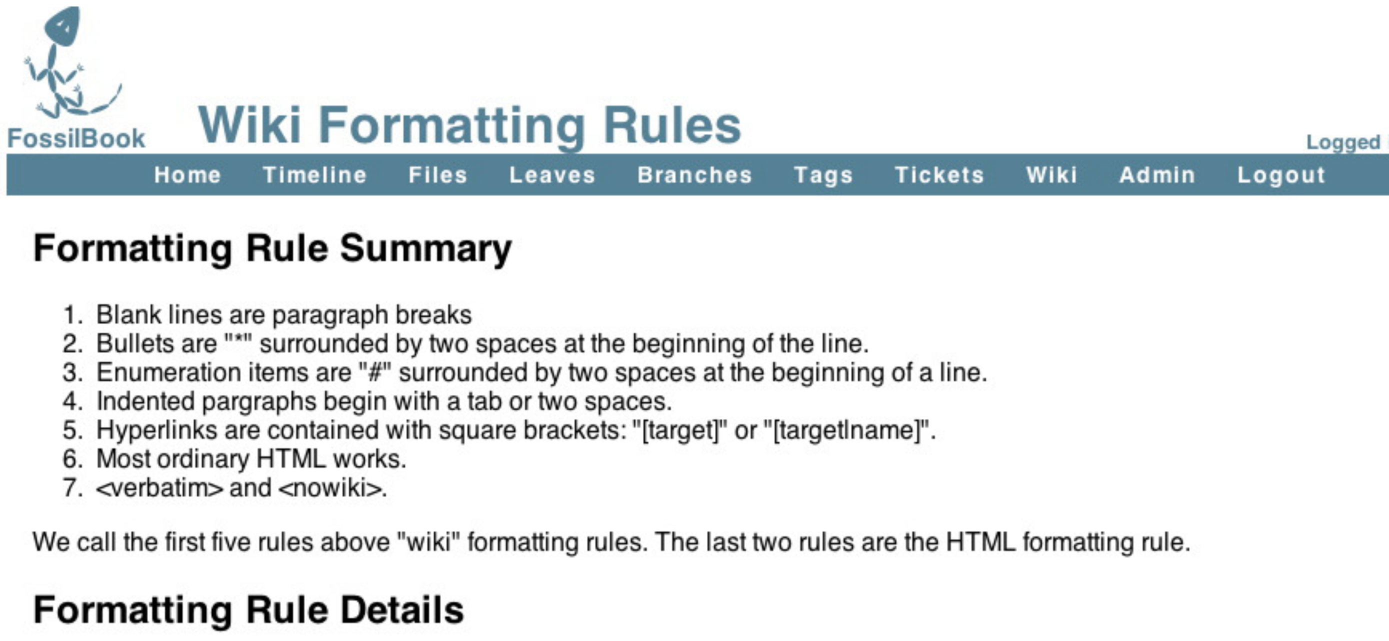
- FossilBook wiki home page.
- Recent changes to wiki pages.
- Formatting rules for wiki.
- Use the Sandbox to experiment.
- Create a new wiki page.
- List of All Wiki Pages available on this server.
- Search wiki titles:

Fossil version [c56af61e5e] 2010-04-22

Figure 21: Wiki controls

These are the controls that let you control and modify the wiki. Most important for now is the Formatting rules link. This link takes you to a page that describes what you can do to format a wiki page. If you just type text on a page it will appear but be formatted by your browser. You can type HTML commands to control this formatting. It's worth your time to carefully read this page and note what you can and cannot do. The page just lists the valid HTML commands, and if you don't know what they mean I would suggest you find a page like this `http://www.webmonkey.com/2010/02/html_cheatsheet/` and keep it handy.

Besides the HTML markup the most powerful command for the Wiki is [page] where it links to a new page. This is how you add pages and how you build your web site of documentation for the repository.



Formatting Rule Summary

1. Blank lines are paragraph breaks
2. Bullets are "*" surrounded by two spaces at the beginning of the line.
3. Enumeration items are "#" surrounded by two spaces at the beginning of a line.
4. Indented pargraphs begin with a tab or two spaces.
5. Hyperlinks are contained with square brackets: "[target]" or "[target|name]".
6. Most ordinary HTML works.
7. <verbatim> and <nowiki>.

We call the first five rules above "wiki" formatting rules. The last two rules are the HTML formatting rule.

Formatting Rule Details

Figure 22: Wiki Formatting

I now begin work. What I want to do is change the home page to be non-empty and also put a link on the home page to the PDF of this book. In Figure 21 on the preceding page I click on the first item, the FossilBook home page. This takes me to the home page again but now I have an Edit option. We also have a History option so I could look at older versions of the page.

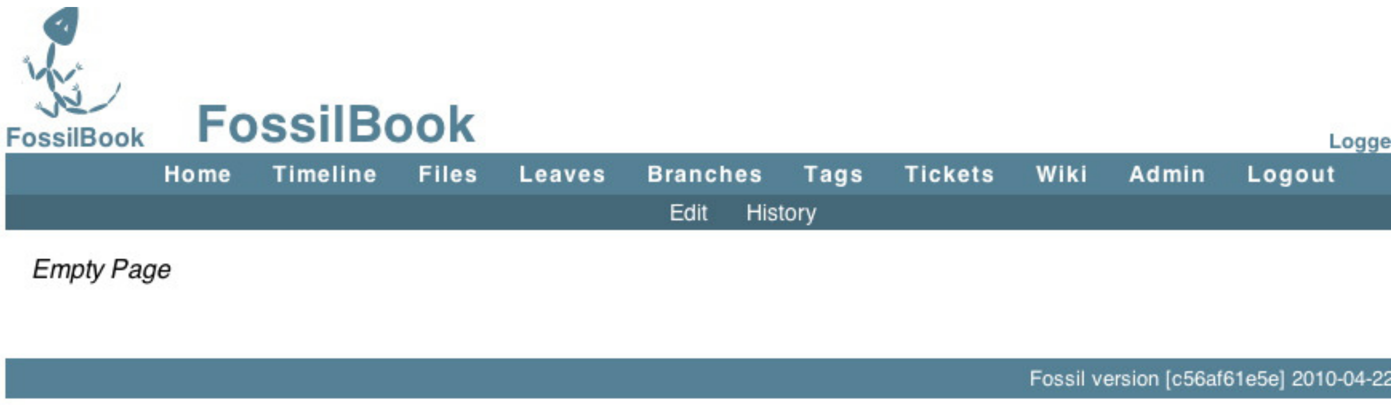
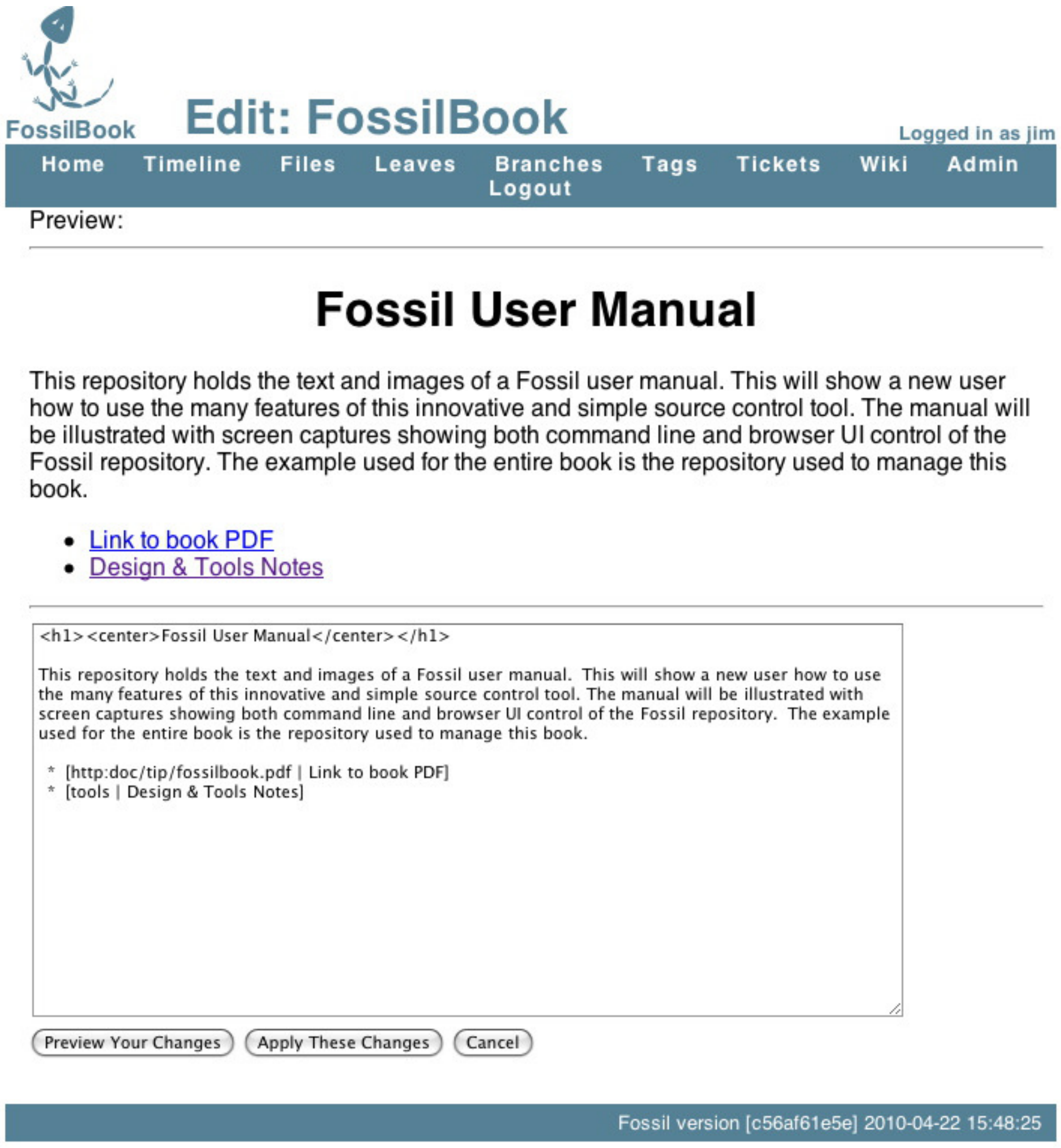


Figure 23: Home page with edit

This shows my initial edit and a preview:


FossilBook
Edit: FossilBook
Logged in as jim
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout
Preview:

Fossil User Manual

This repository holds the text and images of a Fossil user manual. This will show a new user how to use the many features of this innovative and simple source control tool. The manual will be illustrated with screen captures showing both command line and browser UI control of the Fossil repository. The example used for the entire book is the repository used to manage this book.

- Link to book PDF
- Design & Tools Notes

```
<h1><center>Fossil User Manual</center></h1>

This repository holds the text and images of a Fossil user manual.  This will show a new user how to use
the many features of this innovative and simple source control tool. The manual will be illustrated with
screen captures showing both command line and browser UI control of the Fossil repository.  The example
used for the entire book is the repository used to manage this book.

  *  [http:doc/tip/fossilbook.pdf | Link to book PDF]
  *  [tools | Design & Tools Notes]
```

Figure 24: Initial Home page

The bottom section is an edit window where I type things I want displayed and the top is a preview of what the page will be. As you can see I typed some simple HTML to make a large and centered title. The body of the text I just typed and as you see the browser fits the text to the screen. You can have multiple paragraphs by just putting blank lines between your text. Next I wanted a bulleted list and this is done by typing two spaces, a '*' then two more spaces. On each of these lines I have a link to a new (not yet created page). If you look I put these in the form [<new page> | <title>]. This way I can have a long string that describes the link but have a nice short (no embedded spaces) page name.

One mistake I usually make at this point is to click one of those new links which takes me to a new blank page. **OOPS**, if I have not saved this page yet then I find I've lost my changes so far.

OK, I will save my changes and then go to the new pages. I am doing some complicated things here. The first link is to the book PDF. This will be a file I create in the repository. The LyX program I'm using creates the PDF. I will do that, save it, and add it to the repository. But I don't want to link to a static file, that is I want the most current version of the PDF, the one I save each time I quit for the day. To do this we have to put in a link that looks like this:

```
[http:doc/tip/FossilBook.pdf | Book (pdf) ]
```

This is a special link the Fossil wiki understands, **doc** says this is documentation. **tip** says use the most current version; you could put a version link here. And finally since I am going to put the book PDF at the top level I only need the file name. If it was in a subdirectory I would have to say **doc/tip/subdir/filename.**

The second link is just to a regular page and I can just edit that one just like I did this the main page.

2.6.1 Wiki Summary

- Format your text using HTML commands like <h1>Title</h1> for page headings
- Create and link pages using [<page> | <Link text>]
- The page designation http:doc/tip/<document path relative to repository> will display any document in the repository that your browser can handle (i.e. pdf, http etc)
- Never click on a link till AFTER you have saved the page

3 Multiple Users

3.1 Introduction

In the previous chapter I went through using Fossil with just one user (me). In this chapter we will get into using it with multiple users. Thanks to Fossil's distributed design once the set up is done using it is not much different than the single user case with Fossil managing automatically the multiple user details.

3.2 Setup

In the previous chapter the Fossil repository was a file on our system and we did commits to it and pulled copies of the source from it. Fossil is a distributed source control system; what this means is that there is a master repository in a place that all users can access. Each user has their own "cloned" copy of the repository and Fossil will automatically synchronize the users repository with the master. From a each user's perspective you have your local repository and work with it using the same commands shown in Chapter 2. It's just that now Fossil will keep your repository in sync with the master.

3.2.1 Server Setup

I have the FossilBook.fossil repository and now have to put it in place so multiple users can access it. There are two ways, the first is using fossil's built in webserver to host the file and the second is using the operating systems supported web server (if present) and a cgi type access.

3.2.1.1 Self hosted This is quite simply the easiest way to do it. The downside is that you are responsible for keeping the machine available and the webserver up. That is, don't turn the machine off when you quit for the day or some other user is going to be upset. All I have to do is this:

```
[Pandora-2:/Users/jschimpf/Public] jim% cd FOSSIL/
[Pandora-2:jschimpf/Public/FOSSIL] jim% fossil ui -port 8081 FossilBook.fossil &
[1] 310
[Pandora-2:jschimpf/Public/FOSSIL] jim%
```

Figure 25: Self-hosted Fossil repository

This is on UNIX system, the "&" at then end of the second line runs the fossil webserver in the background. If I know this machine has an IP address of 192.168.1.200 then from any other machine in the network I can say:

http://192.168.1.200:8081 to the browser and I can access the Fossil web server.

As you can see this is simple and works on any system that runs Fossil. As long as you carefully make sure it's always running and available for others this can be a very easy way to make the master repository available.

The problems with this method are:

1. If you have multiple repositories you have to use the **server** not the **ui** command, have all your repositories in the same directory, and have them all use the extension .fossil.
2. If the machine goes off line (i.e. for OS update) or other reason it might not automatically restart the Fossil servers.
3. Backup of the repositories might be not be done.

This method does work, and if you only have one repository and a diligent owner of the master machine, it will work and work well.

3.2.1.2 Server hosted If you have a server type machine available (i.e., a Linux or UNIX box) that is running Apache or a Windows machine running IIS you can let it be the webserver for your repository. This has a number of advantages: this machine will be up all the time, it will probably be automatically backed up, and it can easily support multiple Fossil repositories.

I am not going into how to set up the webserver or how to enable (Common Gateway Interface) CGI. See the following sites.

- For IIS see here `http://www.boutell.com/newfaq/creating/iiscgihowto.html` and just ensure that your fossil.exe is in the path to be run by the cgi script.
- For Apache see here `http://httpd.apache.org/docs/2.0/howto/cgi.html` and ensure you know the directory where the CGI scripts are stored.

If you are not in control of the webserver you will need the help of the server admin to enable CGI and to copy your CGI scripts to correct location.

3.2.1.3 CGI Script for hosted server If we assume an Apache server and, in my case, the cgi directory path is /Library/Webserver/CGI-Executables, then we have to write a script of the form:

```
#!<Fossil executable location>
repository: <Fossil repository location>
```

Figure 26: Fossil CGI script

and put it into the cgi script directory. I have put my Fossil executable into /usr/local/bin and I am putting my Fossil shared repository into /Users/Shared/FOSSIL. That is I am using the same Fossil repository as the previous example (see 3.2.1.1 on the previous page) but now it will be references by Apache instead of the Fossil web server. My script then becomes:

```
#!/usr/local/bin/fossil
# Put the book repository on the web
repository: /Users/Shared/FOSSIL/Fossilbook.fossil
```

Figure 27: My Fossil CGI script

After making the script I then copy it to the CGI directory and allow anyone to execute it.

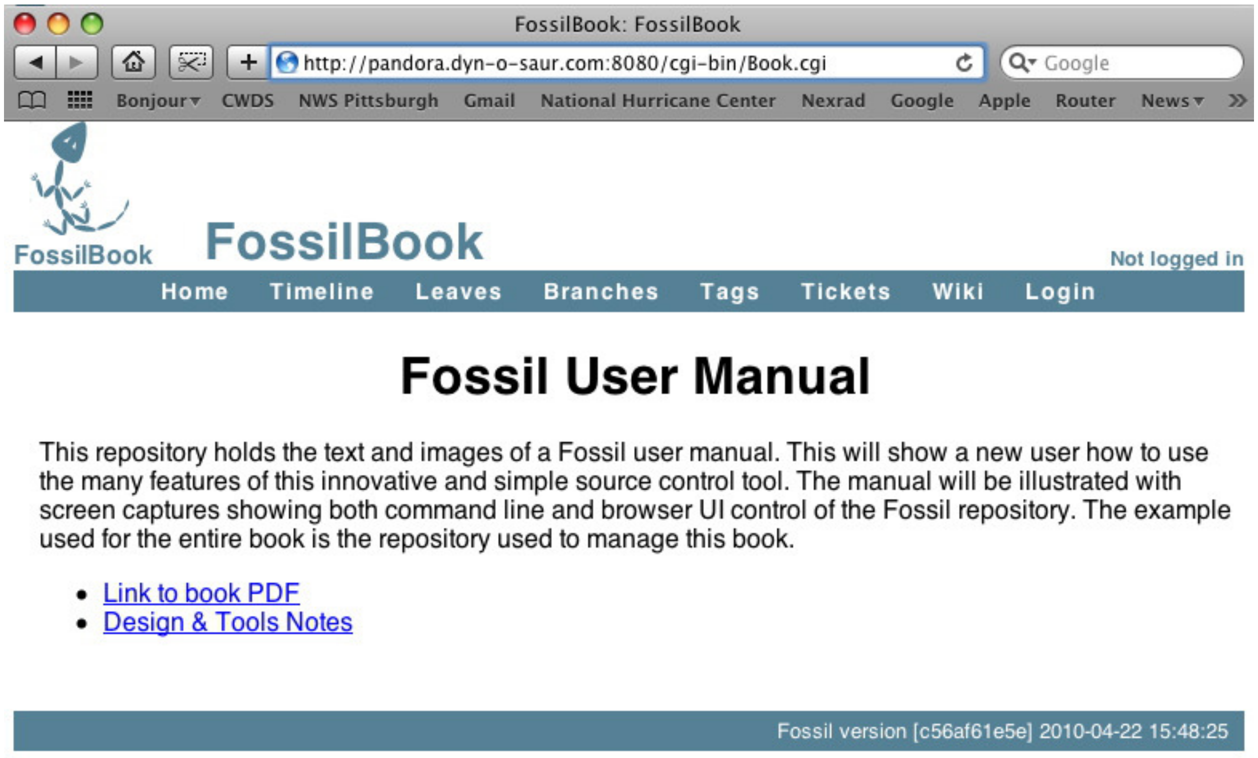
```
sudo cp Book.cgi /Library/Webserver/CGI-Executables/Book.cgi
sudo chmod a+x /Library/Webserver/CGI-Executables/Book.cgi
```

Figure 28: Copying script into place

After doing this the Apache server will execute the CGI script and it will run Fossil and bring up the repository website.

3.2.2 The test (either self hosted or server hosted)

If all is in place then I should be able to access the webserver and get to this:



Fossil User Manual

This repository holds the text and images of a Fossil user manual. This will show a new user how to use the many features of this innovative and simple source control tool. The manual will be illustrated with screen captures showing both command line and browser UI control of the Fossil repository. The example used for the entire book is the repository used to manage this book.

- Link to book PDF
- Design & Tools Notes

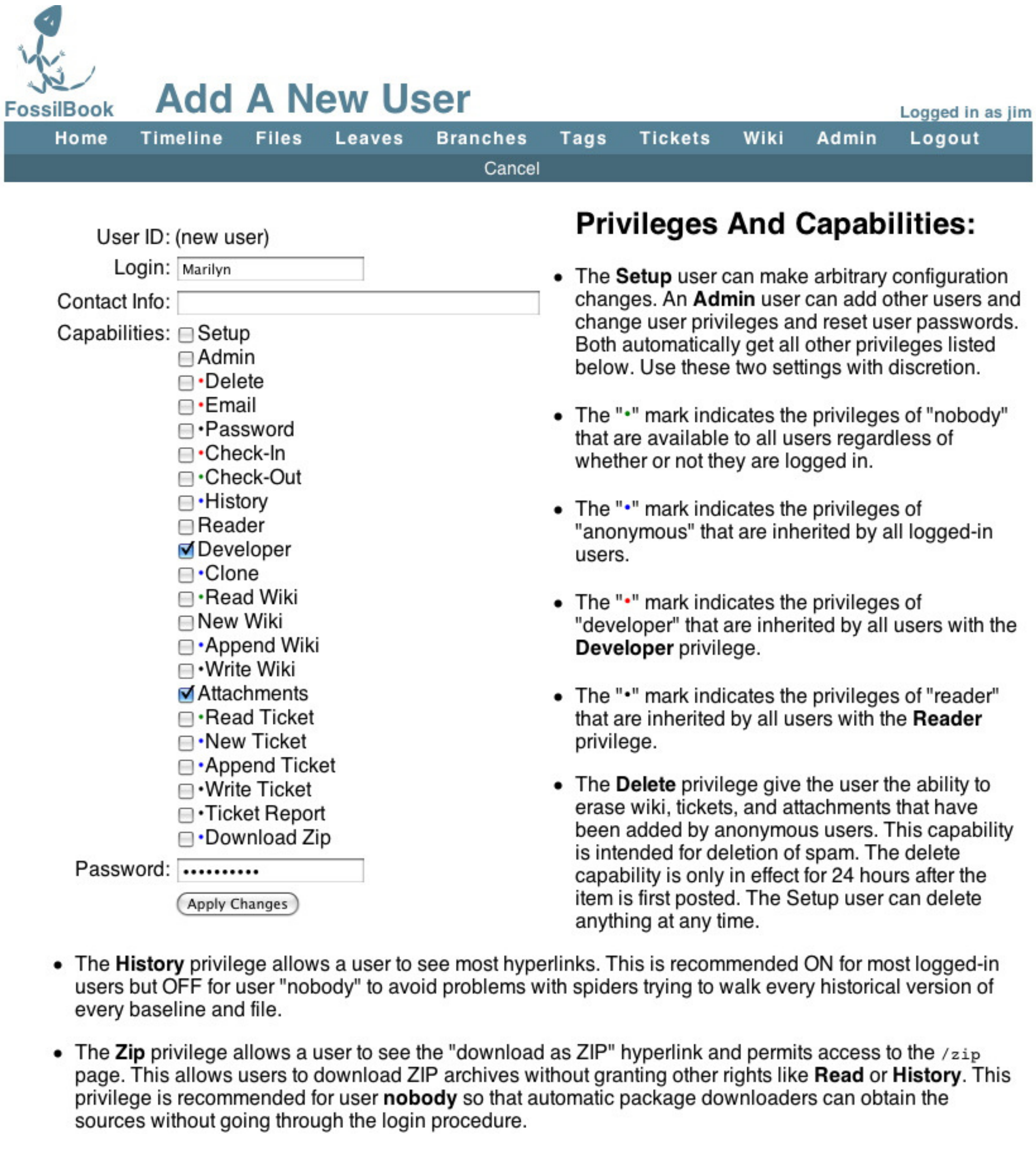
Fossil version [c56af61e5e] 2010-04-22 15:48:25

Figure 29: Web access to Fossil CGI hosted site

3.3 User Accounts

Serving a repository, either self hosting or the more complicated CGI method gets you to the same place as shown in Figure 29. Now I have to set up user accounts for the other contributors to this book. Remember Fossil has automatically created an Anonymous user (see Figure 8 on page 13) thus others can access the site in a limited way, that is they can download the book but cannot commit changes. In this case I want to create a new account (Marilyn) that can make changes and commit changes as she is my editor.

To accomplish all this first I have to login by going to the log in page and entering my ID (jim) and my password. Now since I'm super-user I then go back to the User-Configuration page, Figure (8) and add a new user:



User ID: (new user)
Login: Marilyn
Contact Info:
Capabilities:
- [ ] Setup
- [ ] Admin
- [ ] •Delete
- [ ] •Email
- [ ] •Password
- [ ] •Check-In
- [ ] •Check-Out
- [ ] •History
- [ ] Reader
- [✓] Developer
- [ ] •Clone
- [ ] •Read Wiki
- [ ] New Wiki
- [ ] •Append Wiki
- [ ] •Write Wiki
- [✓] Attachments
- [ ] •Read Ticket
- [ ] •New Ticket
- [ ] •Append Ticket
- [ ] •Write Ticket
- [ ] •Ticket Report
- [ ] •Download Zip

Password: ••••••••••

Privileges And Capabilities:

- The **Setup** user can make arbitrary configuration changes. An **Admin** user can add other users and change user privileges and reset user passwords. Both automatically get all other privileges listed below. Use these two settings with discretion.
- The "•" mark indicates the privileges of "nobody" that are available to all users regardless of whether or not they are logged in.
- The "•" mark indicates the privileges of "anonymous" that are inherited by all logged-in users.
- The "•" mark indicates the privileges of "developer" that are inherited by all users with the **Developer** privilege.
- The "•" mark indicates the privileges of "reader" that are inherited by all users with the **Reader** privilege.
- The **Delete** privilege give the user the ability to erase wiki, tickets, and attachments that have been added by anonymous users. This capability is intended for deletion of spam. The delete capability is only in effect for 24 hours after the item is first posted. The Setup user can delete anything at any time.
- The **History** privilege allows a user to see most hyperlinks. This is recommended ON for most logged-in users but OFF for user "nobody" to avoid problems with spiders trying to walk every historical version of every baseline and file.
- The **Zip** privilege allows a user to see the "download as ZIP" hyperlink and permits access to the `/zip` page. This allows users to download ZIP archives without granting other rights like **Read** or **History**. This privilege is recommended for user **nobody** so that automatic package downloaders can obtain the sources without going through the login procedure.

Figure 30: New Editor user

Since she is going to be an editor, this will be similar to a developer if we were doing code, so I picked the Developer privilege level. This way she can get the repository, check-in, check-out, and write and update tickets. I also added the attachments since she might need that to put on an image or other comment on actions she is doing. I also gave her a password so her access can be secured.

I could add other users at this point but don't need any others for this project, but you can see how easily this can be done. When you assign the user privileges just read carefully and don't give them any more than you think they need. If they have problems, you can easily modify their account in the future.

3.4 Multiple User Operation

With the server set up and the user established the next thing to do is clone the repository. That is make copy from the webserver repository to my local machine. Once that is done this local repository uses the same commands and is very much like single user use discussed in Section 2 on page 8. Fossil will synchronize your local repository with the one on the server.

3.4.1 Cloning

To clone a Fossil repository you have to know four things:

1. It's web address, for our repository it is **http://pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi**
2. Your account name in my case it's **jim**
3. Your password (which I'm keeping to myself thank you...)
4. The local name of the repository, in this case I'm calling it Fossilbook.fossil

You then go to where you want to keep the Repository (in my case the FOSSIL directory) and use the clone command:

```
[Pandora-2:jschimpf/Public/FOSSIL] fossil clone http://jim:<passwd>@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi FossilBook.fossil
### NOTE: <passwd> - Stands in for real password
                Bytes      Cards  Artifacts     Deltas
Send:              49          1          0          0
Received:         120          2          0          0
Send:             625         25          0          0
Received:        4704         72          0          0
Send:            3104         72          0          0
Received:     5129052        131         10          5
Send:            2399         51          0          0
Received:     4381170        116         22         28
Total network traffic: 4117 bytes sent, 6913068 bytes received
Rebuilding repository meta-data...
65 (100%)...
project-id: 2b0d35831c1a5b315d74c4fd8d532b100b822ad7
server-id:  3e67da6d6212494456c69b1c5406a277d7e50430
admin-user: jim (password is "d07222")
[Pandora-2:jschimpf/Public/FOSSIL] jim%
```

Figure 31: Clone command

At this point I can go through the steps outlined in Section 2 on page 8 to set my user password and then open the Fossil Repository on a working directory.

Now that I've moved everything to the new cloned repository I do a check in a the end of the day which looks like this:

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "Moved to clone repository"
Autosync: http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
Bytes Cards Artifacts Deltas
Send: 130 1 0 0
Received: 2990 65 0 0
Total network traffic: 334 bytes sent, 1876 bytes received
New_Version: 158492516c640d055bc0720684a05e797b88165a
Autosync: http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
Bytes Cards Artifacts Deltas
Send: 618798 74 1 2
Received: 3222 70 0 0
Send: 268138 73 1 0
Received: 3221 70 0 0
Send: 3977 72 0 1
Received: 3220 70 0 0
Total network traffic: 457995 bytes sent, 6011 bytes received
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 32: Cloned repository checkin

As you see the files were committed locally and then the local repository was automatically synchronized with the repository on the server.

3.4.2 Keeping in sync

After doing all the setup described above I now have a distributed source control system. My co-worker, Marilyn has also cloned the repository and begun work. She is editing the book correcting my clumsy phrasing and fixing spelling mistakes. She is working independently and on the same files I use. We must use Fossil to prevent us from both modifying FossilBook.lyx but in different ways. Remember Fossil has no file locking, there is nothing to prevent her from editing and changing the file while I work on it.

This is where we both must follow procedures to prevent this sort of problem. Even though she edits files I cannot see the changes till they are committed. Two different versions of the same file won't be a problem till I try to commit with my version and her version is in the current leaf.

There are two problems:

1. Before I do any work I must be sure I have the current versions of all the files.
2. When I commit I must make sure what I am committing has only my changes and is not stepping on changes she has done.

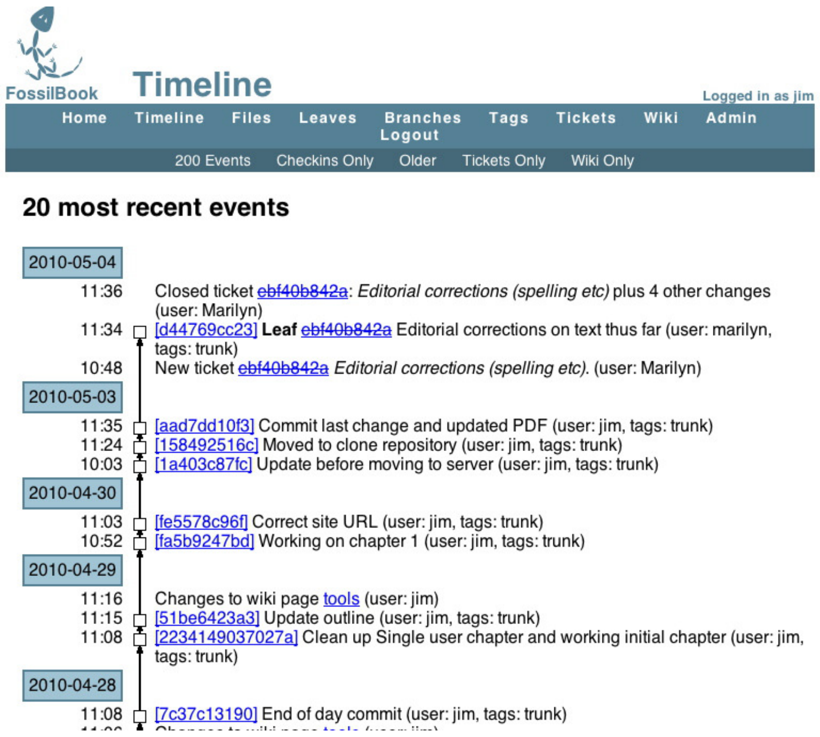
The first is pretty obvious, make sure you have the latest before you do anything. We do that with the update command. In Figure 32 I had done my latest check in. Before starting any more work I

```
[Pandora-2:jschimpf/Public/FossilBook] jim% **fossil update trunk**
Autosync: http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
Bytes Cards Artifacts Deltas
Send: 130 1 0 0
Received: 3588 78 0 0
Send: 365 6 0 0
Received: 136461 83 2 3
Total network traffic: 796 bytes sent, 131582 bytes received
UPDATE fossilbook.lyx
UPDATE fossilbook.pdf
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 33: Update action

should ensure that Marilyn hasn't checked in something else. I could check the timeline but instead I'll do an update to my repository and source files. When I do the update I specify it should be updated from the **trunk.** This ensures I get it from the latest and greatest not some branch.

Ah ha ! Marilyn has been at work and updated the book source and pdf. If I check the timeline from the webserver I see she has even documented it:



20 most recent events

2010-05-04
11:36 Closed ticket ~~ebf40b842a~~: *Editorial corrections (spelling etc)* plus 4 other changes (user: Marilyn)
11:34 [d44769cc23] **Leaf** ~~ebf40b842a~~ Editorial corrections on text thus far (user: marilyn, tags: trunk)
10:48 New ticket ~~ebf40b842a~~ *Editorial corrections (spelling etc).* (user: Marilyn)

2010-05-03
11:35 [aad7dd10f3] Commit last change and updated PDF (user: jim, tags: trunk)
11:24 [158492516c] Moved to clone repository (user: jim, tags: trunk)
10:03 [1a403c87fc] Update before moving to server (user: jim, tags: trunk)

2010-04-30
11:03 [fe5578c96f] Correct site URL (user: jim, tags: trunk)
10:52 [fa5b9247bd] Working on chapter 1 (user: jim, tags: trunk)

2010-04-29
11:16 Changes to wiki page tools (user: jim)
11:15 [51be6423a3] Update outline (user: jim, tags: trunk)
11:08 [2234149037027a] Clean up Single user chapter and working initial chapter (user: jim, tags: trunk)

2010-04-28
11:08 [7c37c13190] End of day commit (user: jim, tags: trunk)

Figure 34: Updated Timeline

Now I know I have the current state of the world and I can proceed to add in new sections.

3.4.3 Complications

In 3.4.2 on page 33 the second case is much harder. In this case I have diligently done my fossil update and started working. In the mean time Marilyn has also done her update and also started working. Now she is done and checks in her changes. I obviously don't know this since I am happily working on my changes. What happens next....

```
[Pandora-2:jschimpf/Public/FossilBook] jim%fossil commit -m "Commit that might fork"
Autosync: http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
Bytes Cards Artifacts Deltas
Send: 130 1 0 0
Received: 4002 87 0 0
Send: 365 6 0 0
Received: 110470 92 2 3
Total network traffic: 797 bytes sent, 104567 bytes received
fossil: would fork. "update" first or use -f or --force.
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 35: Forking commit

Ah ha, that very thing has happened and fossil warned me that my copy of the file differs from the master copy. If I did a –force then the repository would generate a fork and Marilyn's future commits would be to her fork and my commits would be to mine. That would not be what we want since I want her to edit my copy of the book.

The next step would be to do as Fossil says and do an update. At this point you have to be careful since blindly updating the changed files could overwrite the stuff I've just done. So we do a trial update by using the -n and -v options to say"do a dry run" and show me the results.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil update -n -v
UNCHANGED Images/Multiple_user/mul_new_user.epsf
UNCHANGED Images/Multiple_user/mul_timeline.epsf
UNCHANGED Images/Multiple_user/test_access.epsf
UNCHANGED Images/Single_user/config_initial.epsf
UNCHANGED Images/Single_user/initial_page.epsf
UNCHANGED Images/Single_user/jim_setup.epsf
UNCHANGED Images/Single_user/ticket_done.epsf
UNCHANGED Images/Single_user/ticket_form.epsf
UNCHANGED Images/Single_user/ticket_initial.epsf
UNCHANGED Images/Single_user/ticket_list.epsf
UNCHANGED Images/Single_user/ticket_submit.epsf
UNCHANGED Images/Single_user/timeline.epsf
UNCHANGED Images/Single_user/timeline_detail.epsf
UNCHANGED Images/Single_user/user_config.epsf
UNCHANGED Images/Single_user/wiki_blankhome.epsf
UNCHANGED Images/Single_user/wiki_formating.epsf
UNCHANGED Images/Single_user/wiki_home.epsf
UNCHANGED Images/Single_user/wiki_homeedit.epsf
UNCHANGED Images/Single_user/wiki_page.epsf
UNCHANGED Layout/fossil.png
UNCHANGED Research/History/CVS-grune.pdf
UNCHANGED Research/History/RCS--A System for Version Control.webloc
UNCHANGED Research/History/SCCS-Slideshow.pdf
UNCHANGED Research/History/VCSHistory - pysync - A Brief History of Version Control Systems - Project Ho
UNCHANGED Research/fossilbib.bib
MERGE fossilbook.lyx
UPDATE fossilbook.pdf
UNCHANGED outline.txt
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 36: Update dry run

That's a little more than I wanted as you can see almost everything is UNCHANGED but it shows that fossilbook.lyx needs a MERGE and fossilbook.pdf needs an UPDATE. This is what I should expect, Marilyn has done edits to the fossilbook.lyx file and so have I so we have to merge the changes. But she has also updated the fossilbook.pdf which I have not. Before we go on if you are running on Linux or UNIX you can simplify this dry run by doing:

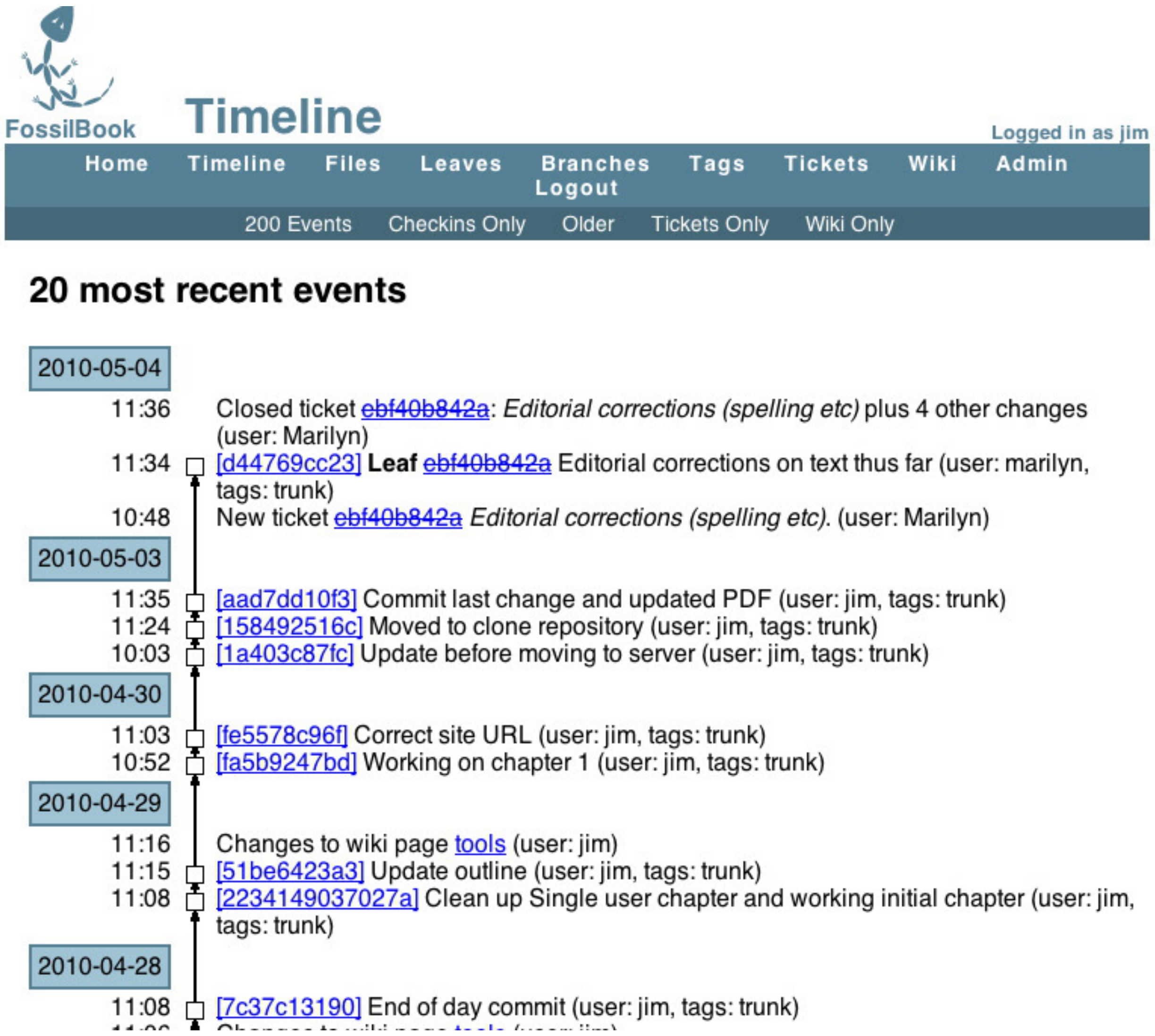
```
[Pandora-2:jschimpf/Public/FossilBook] jim%fossil update -n -v | grep -v UNCHANGED
MERGE fossilbook.lyx
UPDATE fossilbook.pdf
```

Figure 37: Update dry run, shorter

By using the pipe and grep I can eliminate all those extra UNCHANGED lines.

3.4.4 Fixing the Update file

First we fix the easy file, the fossilbook.pdf I can just update by itself so it matches the current repository. It doesn't need merged so just replace it. Before I do that I have to look at the repository time line



20 most recent events

2010-05-04
11:36 Closed ticket ~~ebf40b842a~~: *Editorial corrections (spelling etc)* plus 4 other changes (user: Marilyn)
11:34 [d44769cc23] **Leaf** ~~ebf40b842a~~ Editorial corrections on text thus far (user: marilyn, tags: trunk)
10:48 New ticket ~~ebf40b842a~~ *Editorial corrections (spelling etc).* (user: Marilyn)

2010-05-03
11:35 [aad7dd10f3] Commit last change and updated PDF (user: jim, tags: trunk)
11:24 [158492516c] Moved to clone repository (user: jim, tags: trunk)
10:03 [1a403c87fc] Update before moving to server (user: jim, tags: trunk)

2010-04-30
11:03 [fe5578c96f] Correct site URL (user: jim, tags: trunk)
10:52 [fa5b9247bd] Working on chapter 1 (user: jim, tags: trunk)

2010-04-29
11:16 Changes to wiki page tools (user: jim)
11:15 [51be6423a3] Update outline (user: jim, tags: trunk)
11:08 [2234149037027a] Clean up Single user chapter and working initial chapter (user: jim, tags: trunk)

2010-04-28
11:08 [7c37c13190] End of day commit (user: jim, tags: trunk)

Figure 38: Current Timeline

I see that the current **Leaf** is [d44769cc23] and it is tagged as **trunk**. I want to update the fossil-book.pdf from there. So I say:

```
[Pandora-2:jschimpf/Public/FossilBook] jim%fossil update trunk fossilbook.pdf
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        4002         87          0          0
Total network traffic: 334 bytes sent, 2412 bytes received
UPDATE fossilbook.pdf
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 39: Update fossilbook.pdf

and it's done.

3.4.5 Fixing the Merge file

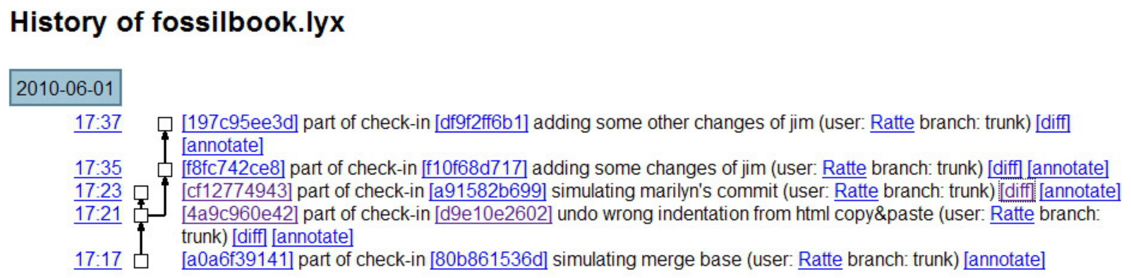
We can use the tools built into Fossil. In this case noticing that commit will cause a fork Jim will use the -force option to cause the fork and will handle the merge later.

```
E:\Profile\Ratte\data\organize\fossil-w32\fossil-book>fossil commit -m "adding some changes of jim"
fossil: would fork.  "update" first or use -f or --force.

E:\Profile\Ratte\data\organize\fossil-w32\fossil-book>fossil commit -f -m "adding some other changes of
New_Version: df9f2ff6b14ef65a9dd2162f8bd20c78e1628165
```

Figure 40: Forcing a commit under Windows

Now the timeline looks like:


History of fossilbook.lyx

2010-06-01
17:37 [197c95ee3d] part of check-in [df9f2ff6b1] adding some other changes of jim (user: Ratte branch: trunk) [diff] [annotate]
17:35 [f8fc742ce8] part of check-in [f10f68d717] adding some changes of jim (user: Ratte branch: trunk) [diff] [annotate]
17:23 [cf12774943] part of check-in [a91582b699] simulating marilyn's commit (user: Ratte branch: trunk) [diff] [annotate]
17:21 [4a9c960e42] part of check-in [d9e10e2602] undo wrong indentation from html copy&paste (user: Ratte branch: trunk) [diff] [annotate]
17:17 [a0a6f39141] part of check-in [80b861536d] simulating merge base (user: Ratte branch: trunk) [annotate]

Figure 41: Windows Forked timeline

To remove this fork (i.e. get the changes Marilyn did into the trunk) we use the Fossil merge command. We can use the merge because fossilbook.lyx is a text file and the merge markers are designed to work with text files. If it was a binary file we might have to use an external file or copy and paste between the two file versions using the handler program for the file.

```
E:\Profile\Ratte\data\organize\fossil-w32\fossil-book>fossil merge a91582b699
MERGE fossilbook.lyx
***** 2 merge conflicts in fossilbook.lyx
```

Figure 42: Fossil Merge

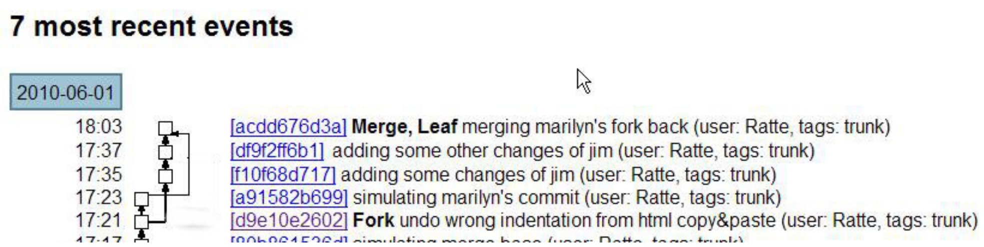
Looking at the file (fossilbook.lyx) in a text editor (not LyX) we find:

```
>>>>>>> BEGIN MERGE CONFLICT
 Thanks to Fossil's distributed design once the set up is done using it
 with multiple users is not much different than the single user case.
 Fossil will automatically manage the most multiple user details.
============================

 Thanks to Fossil's distributed design once the set up is done using is
 not much different than the single user case with Fossil managing automatically
 the multiple user details.
<<<<<<< END MERGE CONFLICT
<Here edit in the changes>
E:\Profile\Ratte\data\organize\fossil-w32\fossil-book>fossil commit -m "merging marilyn's fork back"
New_Version: acdd676d3ab157769496f6845ccc7652985c1d03
```

Figure 43: Text differences

After the commit the timeline shows how the merge brought the fork back into the main trunk. Marilyn will then have to update to this new trunk. (See Section **??** on page ??)


7 most recent events

2010-06-01

```
18:03  [acdd676d3a] Merge, Leaf merging marilyn's fork back (user: Ratte, tags: trunk)
17:37  [df9f2ff6b1] adding some other changes of jim (user: Ratte, tags: trunk)
17:35  [f10f68d717] adding some changes of jim (user: Ratte, tags: trunk)
17:23  [a91582b699] simulating marilyn's commit (user: Ratte, tags: trunk)
17:21  [d9e10e2602] Fork undo wrong indentation from html copy&paste (user: Ratte, tags: trunk)
17:17  [80b861526d] simulating merge base (user: Ratte, tags: trunk)
```

Figure 44: Merged timeline

4 Forks & Branches

4.1 Introduction

This chapter will cover forking and branching in Fossil. Forking is where you unintentionally create two places to check into a repository. Branching is where you intentionally do this because you want to maintain two or more versions of the code in the same repository. We illustrated forking and it's solutions in Section 3.4.3 on page 35. If, instead of fixing (merging) the file then doing the commit, we forced the commit, Fossil would fork the repository.

Forking is something to avoid because it creates two checkin paths for the code. Thus different users will be on different paths and can check in contradictory changes. Branches on the other hand are forks that you desire. These occur when you want to have two different versions of the code base in development at the same time. This was described in 1.1 on page 5 where you have a production version of code under maintenance and a development version both served from the same repository. In this case development changes should only be checked into the development branch. Maintenance changes might have to be checked into both.

Instead of using the book repository for these examples we will use a JSON[1]parser program that has a number of files and documentation. This will make it simpler to illustrate branching and tagging.

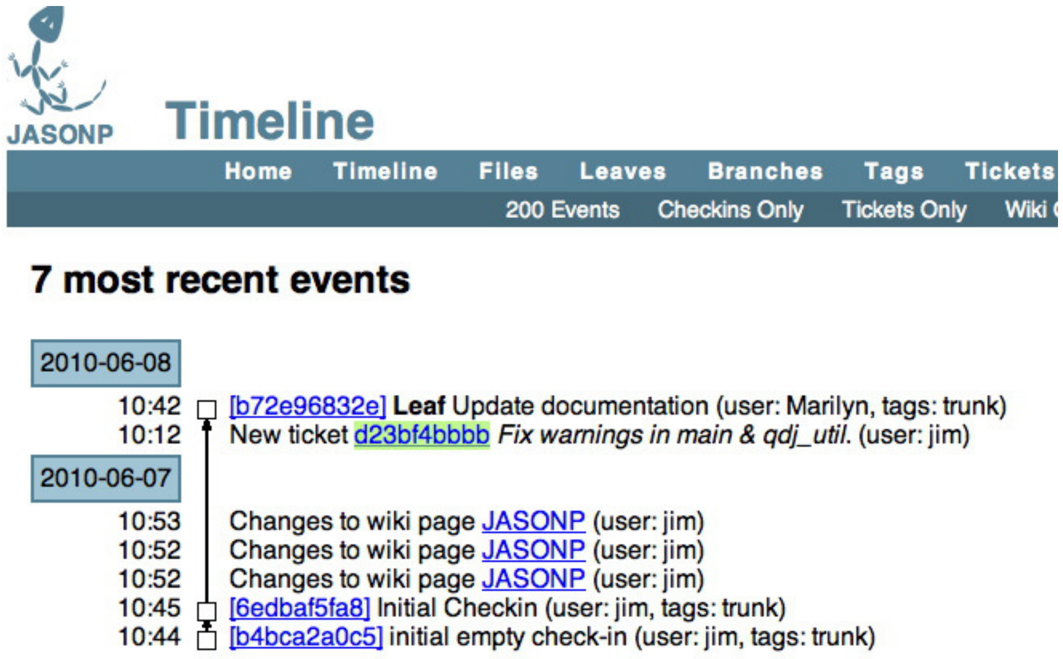
There is a good discussion of these topics on the Fossil Web site http://www.fossil-scm.org/index.html/doc/tip/www/branching.wiki.

4.2 Forks, Branch & Merge

In this case the JSON code has just been placed in Fossil and two developers check out copies to work on. Jim wants to fix a number of compiler warnings that appear and Marilyn wants to fix the documentation. In both cases they proceed as shown in Chapter 3 on page 27. The JSON code has been placed in a distributed location, each of them clones the repository, and opens a working copy of the code.

4.2.1 Marilyn's Actions

She looks through the documentation and finds a number of problems and fixes them (the documentation uses LyX and PDF's). When she is satisfied with what she has done, she checks the current version of the documentation in:



7 most recent events

```
2010-06-08
10:42  [b72e96832e] Leaf Update documentation (user: Marilyn, tags: trunk)
10:12  New ticket d23bf4bbbb Fix warnings in main & qdj_util. (user: jim)
2010-06-07
10:53  Changes to wiki page JASONP (user: jim)
10:52  Changes to wiki page JASONP (user: jim)
10:52  Changes to wiki page JASONP (user: jim)
10:45  [6edbaf5fa8] Initial Checkin (user: jim, tags: trunk)
10:44  [b4bca2a0c5] initial empty check-in (user: jim, tags: trunk)
```

Figure 45: Marilyn's work

4.2.2 Jim's Actions

At the same time, Jim gets a working copy of version [6edbaf5fa8] of the code, puts in a ticket [d23bf4bbbb] as shown in Figure 45. After fixing the warnings, Jim is done and goes to commit. He does this AFTER Marilyn has done her commit.

```
551 jsonp> fossil commit -m "[d23bf4bbbb] Remove warnings"
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:         874         19          0          0
Total network traffic: 339 bytes sent, 771 bytes received
fossil: would fork.  "update" first or use -f or --force.
552 jsonp>
```

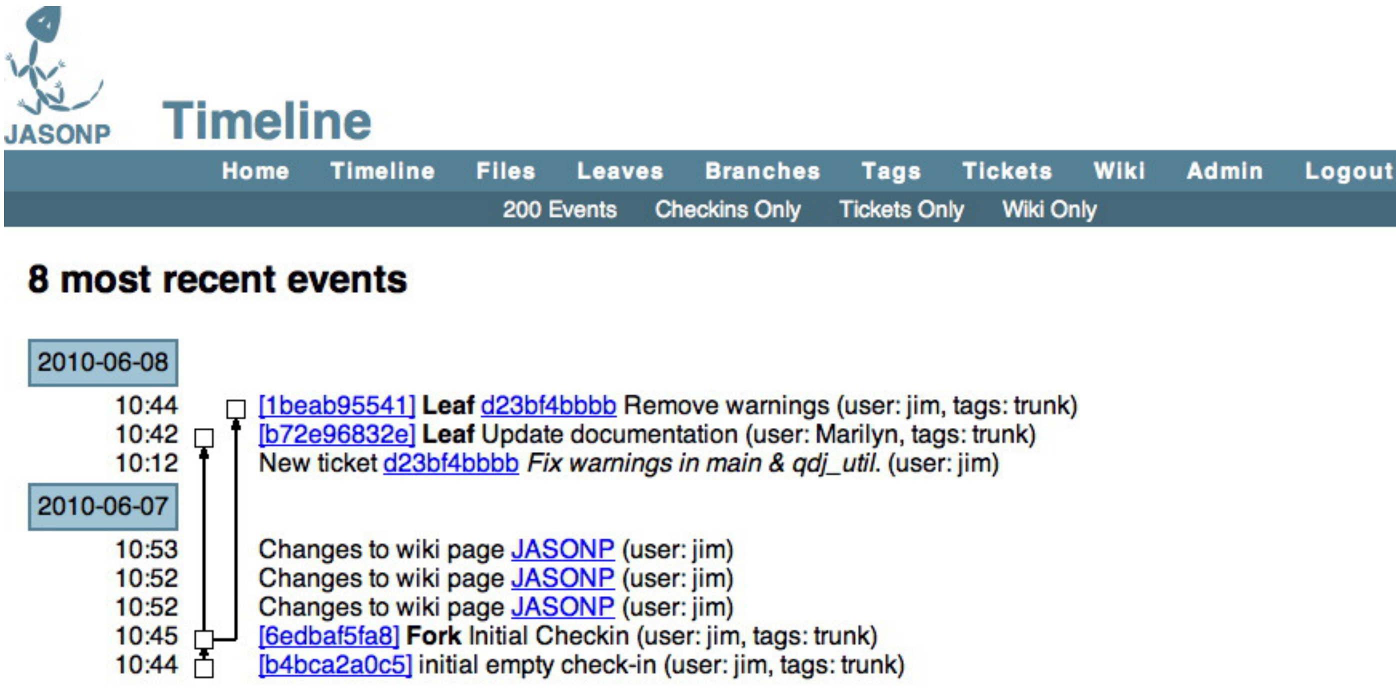
Figure 46: Jim's commit attempt

At this point Fossil recognizes that Marilyn has changed the repository (she updated the documentation) but Jim does not have these changes because he checked out an earlier version of the code. Jim says he **must** get his changes in so he does a FORCE to force fossil to accept the commit.

```
552 jsonp> fossil commit -m "[d23bf4bbbb] Remove warnings" -f
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:         874         19          0          0
Total network traffic: 338 bytes sent, 771 bytes received
New_Version: 1beab955418a942ab9953c4865109ff46cbbd691
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            2646         25          0          4
Received:        1058         23          0          0
Total network traffic: 1498 bytes sent, 864 bytes received
**** warning: a fork has occurred *****
```

Figure 47: Forcing the commit

Looking at the timeline Jim sees this:


JASONP
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin Logout
200 Events Checkins Only Tickets Only Wiki Only

Timeline

8 most recent events

```
2010-06-08
10:44   [1beab95541] Leaf d23bf4bbbb Remove warnings (user: jim, tags: trunk)
10:42   [b72e96832e] Leaf Update documentation (user: Marilyn, tags: trunk)
10:12   New ticket d23bf4bbbb Fix warnings in main & qdj_util. (user: jim)
2010-06-07
10:53   Changes to wiki page JASONP (user: jim)
10:52   Changes to wiki page JASONP (user: jim)
10:52   Changes to wiki page JASONP (user: jim)
10:45   [6edbaf5fa8] Fork Initial Checkin (user: jim, tags: trunk)
10:44   [b4bca2a0c5] initial empty check-in (user: jim, tags: trunk)
```

Figure 48: Repository Fork

Not good, there are two **Leaf**'s and Marilyn would commit code to her fork and Jim would be commuting code to his. So Jim must fix this by merging the code. Jim wants to merge versions [b72e96832e] of Marilyn and his [1beab85441].

4.2.3 Fixing the fork

So Jim who's checked out code is from Leaf [1beab85441] does a merge with Marilyn's leaf [b72e96832e] like so:

```
556 jsonp> fossil merge b72e96832e
UPDATE docs/qdj.lyx
UPDATE docs/qdj.pdf
557 jsonp> fossil status
repository:   /Users/jschimpf/Public/FOSSIL/jsonp.fossil
local-root:   /Users/jschimpf/Public/jsonp/
server-code:  d3e7932b0b0f5e704264ba30adeae14978c08bc6
checkout:     1beab955418a942ab9953c4865109ff46cbbd691 2010-06-08 10:44:56 UTC
parent:       6edbaf5fa8e4d061c2e04e7fd481e7663b090bd3 2010-06-07 10:45:57 UTC
tags:         trunk
UPDATED_BY_MERGE docs/qdj.lyx
UPDATED_BY_MERGE docs/qdj.pdf
MERGED_WITH b72e96832e024f235696dcd6c5d0ddcc2cb38238
```

Figure 49: Merge Operation

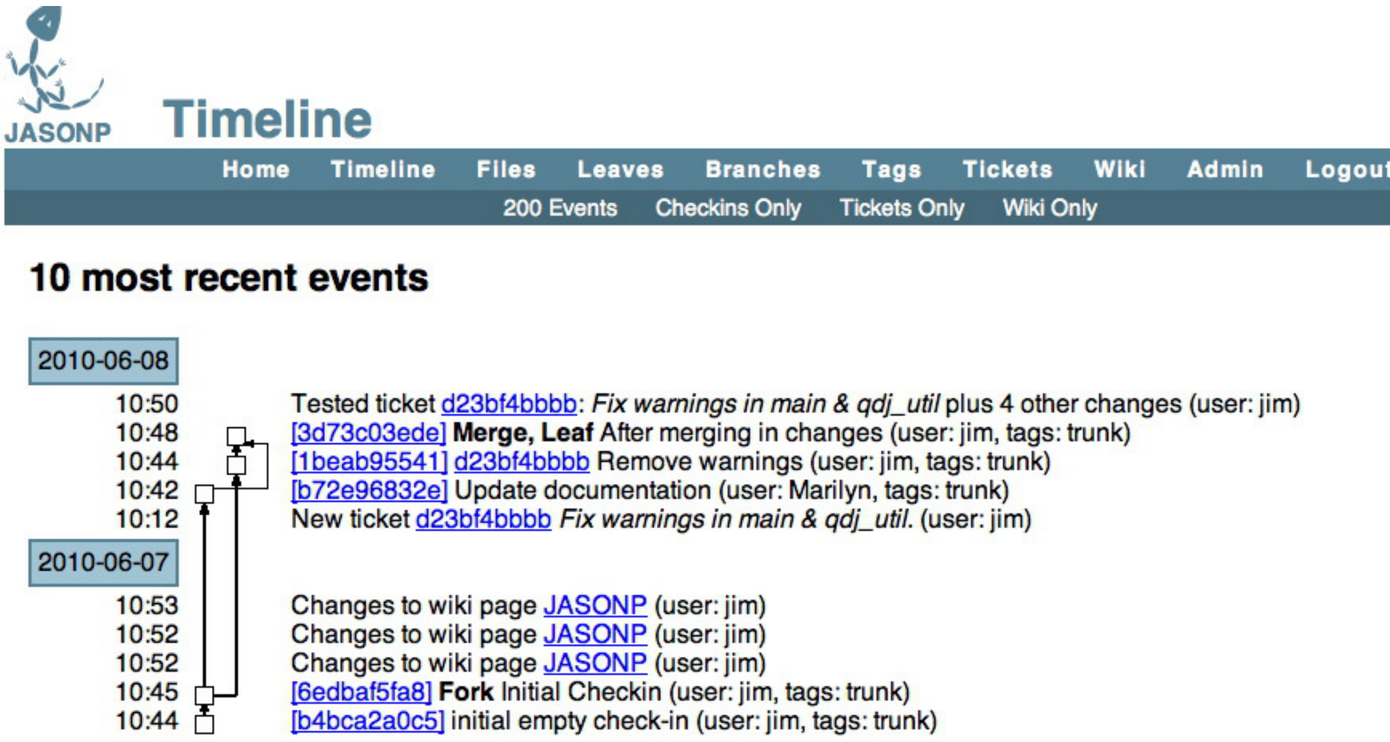
As shown the two documentation files are updated, there are no merge conflicts as Jim didn't touch these files and Marilyn didn't touch the code files.

Next Jim does a commit to make this new merged set of files the new trunk. Remember doing the merge shown in Figure 49 just updates your checked out code and does not change the repository till you check it in.

```
558 jsonp> fossil commit -m "After merging in changes"
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        1058         23          0          0
Total network traffic: 340 bytes sent, 864 bytes received
New_Version: 3d73c03edee33cdc2e1bd8a47de57b7a6b6d880a
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            1737         26          0          1
Received:        1104         24          0          0
Total network traffic: 1101 bytes sent, 888 bytes received
559 jsonp>
```

Figure 50: Commit after merge

When we look at the timeline we have a single leaf for future code commits.



10 most recent events

2010-06-08

10:50 Tested ticket d23bf4bbbb: *Fix warnings in main & qdj_util* plus 4 other changes (user: jim)
10:48 [3d73c03ede] **Merge, Leaf** After merging in changes (user: jim, tags: trunk)
10:44 [1beab95541] d23bf4bbbb Remove warnings (user: jim, tags: trunk)
10:42 [b72e96832e] Update documentation (user: Marilyn, tags: trunk)
10:12 New ticket d23bf4bbbb *Fix warnings in main & qdj_util.* (user: jim)

2010-06-07

10:53 Changes to wiki page JASONP (user: jim)
10:52 Changes to wiki page JASONP (user: jim)
10:52 Changes to wiki page JASONP (user: jim)
10:45 [6edbaf5fa8] **Fork** Initial Checkin (user: jim, tags: trunk)
10:44 [b4bca2a0c5] initial empty check-in (user: jim, tags: trunk)

Figure 51: After merge timeline

The only other thing remaining is that Marilyn does an Update before proceeding so her checked out code matches the repository.

```
WhiteBook:jsonp marilyn$ fossil update
Autosync:  http://Marilyn@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        1150         25          0          0
Send:             412          7          0          0
Received:        3274         31          1          5
Total network traffic: 843 bytes sent, 2709 bytes received
UPDATE json-src/qdj_token.c
UPDATE json-src/qdj_util.c
UPDATE main.c
```

Figure 52: Marilyn's Update

4.2.4 Commands used

- **fossil merge <fork>** Used to merge a fork (specified by hash value) to current check out.

- **fossil update <version>** Used to update current check out to specified version, if version not present use default tag for check out (see fossil status)

4.3 Merge without fork

In this case I will show how to merge in code changes from multiple users without causing a fork. In this case Marilyn has put in a BSD license text into all the code files while Jim is adding a help function to the code. In this case both of them put in tickets saying what they are doing but acting independently.

4.3.1 Check in attempt

Marilyn finished first and checks in her changes. Jim builds, tests and tries to check in his code and gets:

```
502 jsonp> make
/usr/bin/gcc  main.c -c -I. -Ijson-src -o obj/main.o
/usr/bin/gcc  \
obj/main.o\
obj/qdj.o\
obj/qdj_util.o\
obj/qdj_token.o\
-o jsonp
503 jsonp> ./jsonp -v
JSON Test Program Ver: [Jun  9 2010] [10:15:00]
SYNTAX: jsonp -i <json text file> [-v]
-i <json text file>   Show parse of JSON
-v                    Show help
506 jsonp> fossil commit -m "[fed383fa1a] Add help to cmd line"
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        1656         36          0          0
Send:             647         12          0          0
Received:       14039         47          4          7
Total network traffic: 942 bytes sent, 4537 bytes received
fossil: would fork.  "update" first or use -f or --force.
```

Figure 53: Jim's check in attempt

4.3.2 Update

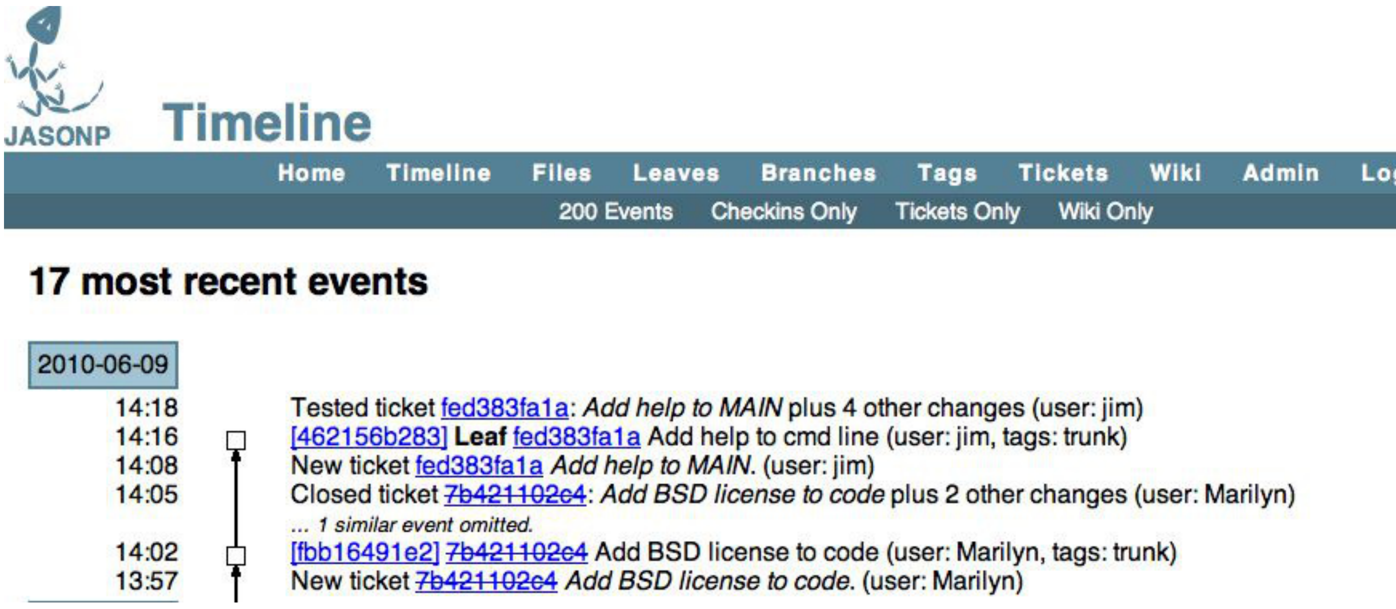
The next action Jim takes is to do the update but without doing changes, using the -n flag which tells it to just show what's going to happen without making any file changes.

```
507 jsonp> fossil update -n
UPDATE json-src/qdj.c
UPDATE json-src/qdj.h
UPDATE json-src/qdj_token.c
UPDATE json-src/qdj_token.h
UPDATE json-src/qdj_util.c
MERGE main.c
```

Figure 54: Update dry run

This shows some files will be updated, i.e. be replaced by new text from the repository. The main.c file will be merged with the version from the repository. That is text from the repository will be mixed with the text from Jim's modified file. Note that it says **MERGE** meaning the two sets of text are a disjoint set. This means the merge can all be done by Fossil with no human intervention.

Jim can just do the update for real then commit the merged files to make a new leaf. So now we have Marilyn's and Jim changes combined in the latest version.


JASONP
Timeline
Home Timeline Files Leaves Branches Tags Tickets Wiki Admin
200 Events Checkins Only Tickets Only Wiki Only

17 most recent events

2010-06-09

```
14:18  Tested ticket fed383fa1a: Add help to MAIN plus 4 other changes (user: jim)
14:16  [462156b283] Leaf fed383fa1a Add help to cmd line (user: jim, tags: trunk)
14:08  New ticket fed383fa1a Add help to MAIN. (user: jim)
14:05  Closed ticket 7b421102c4: Add BSD license to code plus 2 other changes (user: Marilyn)
       ... 1 similar event omitted.
14:02  [fbb16491e2] 7b421102c4 Add BSD license to code (user: Marilyn, tags: trunk)
13:57  New ticket 7b421102c4 Add BSD license to code. (user: Marilyn)
```

Figure 55: Merged repository

4.3.3 Commands used

- **fossil update -n** Does a dry run of an update to show what files will changed.
  - UPDATE Implies file will be replaced by repository file
  - MERGE Implies file will be mixed text from repository and check out

4.4 Branching

4.4.1 Introduction

We have discussed this before but branching is the intention splitting of the code in the repository into multiple paths. This will usually be done with production code where we have maintenance branch and a development branch. The maintenance branch is in use and would get bug fixes based on experience. The development branch would get those changes if applicable plus be modified to add features.

The JSON code parser has been tested and works so will be released to general use. Also we wish to modify it to add support for UTF-8 characters so it matches the JSON standard. The current version just works with ASCII 7 bit characters which is not standard. We wish to split the code into a VER_1.0 branch which is the current code in use and VER_2.0 branch which will add UTF-8 character support.

4.4.2 Branch the repository

Before proceeding we will make sure we have the current trunk code in our check out.

```
[Pandora-2:jschimpf/Public/jsonp] jim% fossil status
repository: /Users/jschimpf/Public/FOSSIL/jsonp.fossil
local-root: /Users/jschimpf/Public/jsonp/
server-code: 90c80f1a2da7360dae230ccec65ff82fe2eb160d
checkout: 462156b283b694af0b99c9b446b64d3f77436fbb 2010-06-09 14:16:42 UTC
parent: fbb16491e2ff9f9ca3a98adffa167de1b6903a44 2010-06-09 14:02:28 UTC
tags: trunk
```

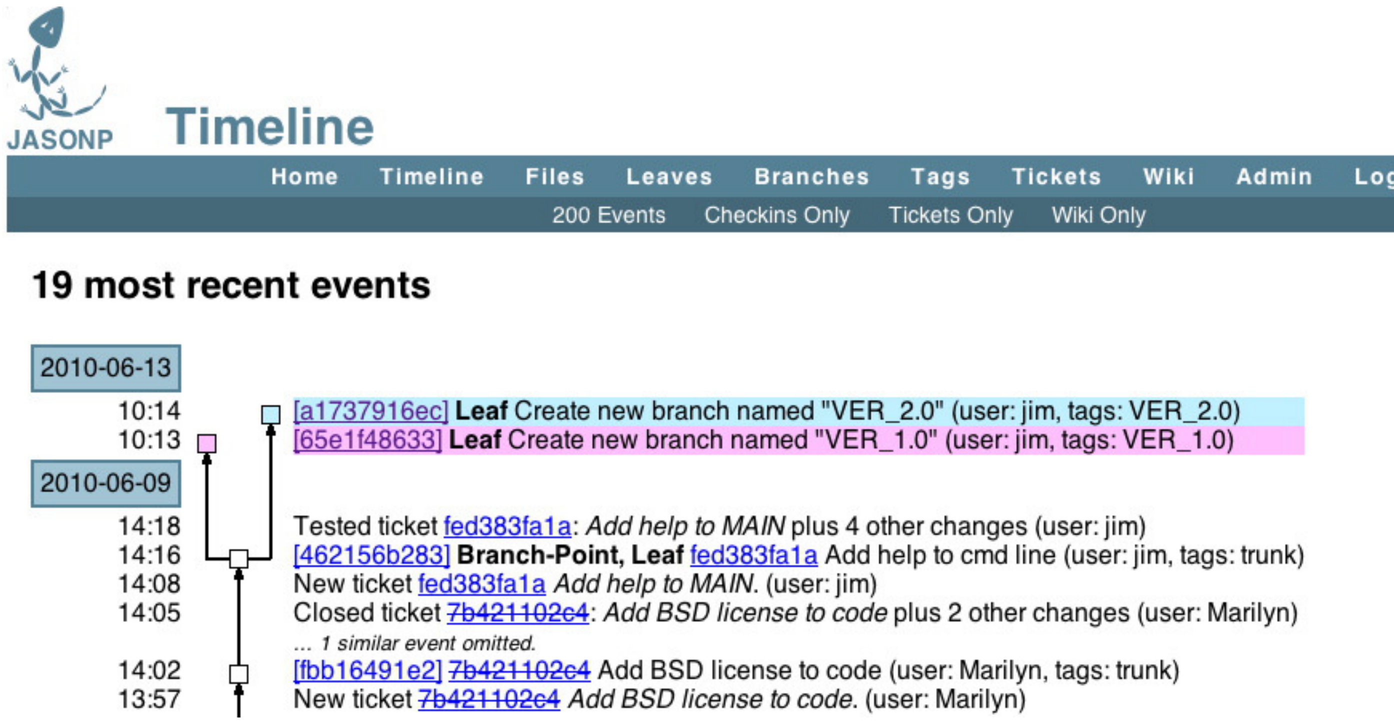
Figure 56: Checking code status

Seeing that matches the latest leaf in the time line we can proceed to branch the code.

```
[Pandora-2:jschimpf/Public/jsonp] jim% fossil branch new VER_1.0 trunk -bgcolor 0xFFC0FF
sh: gpg: command not found
unable to sign manifest.  continue (y/N)? y
New branch: 65e1f48633d691a5ea738cd51ccbf9a581dfb3c7
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            2391         42          0          1
Received:        1840         40          0          0
Total network traffic: 1524 bytes sent, 1272 bytes received
[Pandora-2:jschimpf/Public/jsonp] jim% fossil branch new VER_2.0 trunk -bgcolor 0xC0F0FF
sh: gpg: command not found
unable to sign manifest.  continue (y/N)? y
New branch: a1737916ec2df696a0f3a7e36edf9ba4370e48a7
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            2437         43          0          1
Received:        1886         41          0          0
Total network traffic: 1550 bytes sent, 1271 bytes received
[Pandora-2:jschimpf/Public/jsonp] jim%
```

Figure 57: Branch commands

What was just done. We used the Fossil branch command to create two branches VER_1.0 and VER_2.0 and assigned each of them a color. We can see the timeline is now:



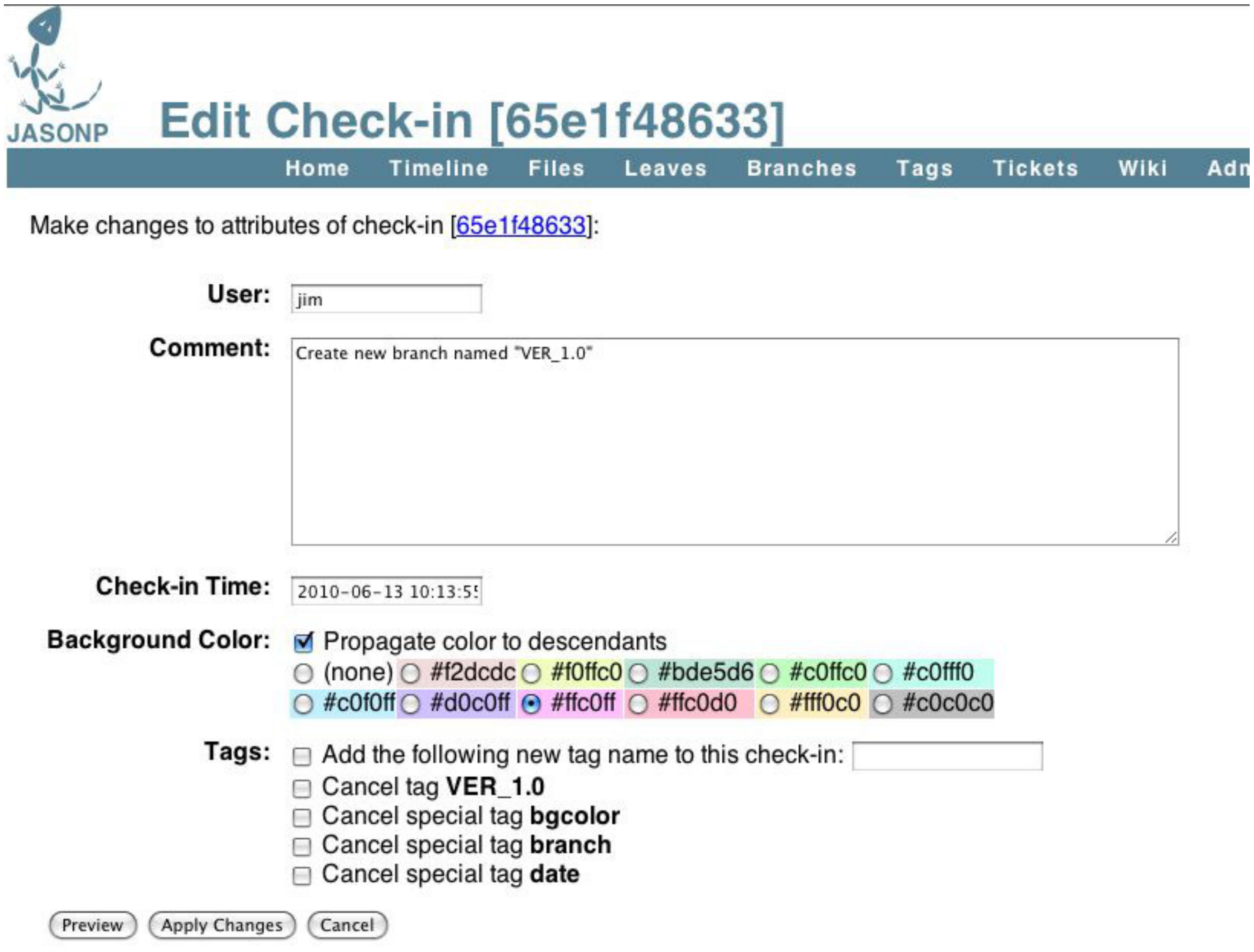
19 most recent events

2010-06-13
10:14 [a1737916ec] **Leaf** Create new branch named "VER_2.0" (user: jim, tags: VER_2.0)
10:13 [65e1f48633] **Leaf** Create new branch named "VER_1.0" (user: jim, tags: VER_1.0)
2010-06-09
14:18 Tested ticket fed383fa1a: *Add help to MAIN* plus 4 other changes (user: jim)
14:16 [462156b283] **Branch-Point, Leaf** fed383fa1a Add help to cmd line (user: jim, tags: trunk)
14:08 New ticket fed383fa1a *Add help to MAIN.* (user: jim)
14:05 Closed ticket ~~7b421102c4~~: *Add BSD license to code* plus 2 other changes (user: Marilyn)
... 1 similar event omitted.
14:02 [fbb16491e2] ~~7b421102c4~~ Add BSD license to code (user: Marilyn, tags: trunk)
13:57 New ticket ~~7b421102c4~~ *Add BSD license to code.* (user: Marilyn)

Figure 58: Branch Timeline

4.4.3 Color Setup

As you see above the two branches have different colors in the timeline. This was due to the **-bgcolor** option added when we created each branch. (See Figure 57). But we want this color to appear on subsequent checkins of each of these branches. To make that happen we have to set the options using the UI and picking a particular leaf on the timeline.



Edit Check-in [65e1f48633]

Make changes to attributes of check-in [65e1f48633]:

User: jim

Comment: Create new branch named "VER_1.0"

Check-in Time: 2010-06-13 10:13:5

Background Color: [✓] Propagate color to descendants

( ) (none) ( ) #f2dcdc ( ) #f0ffc0 ( ) #bde5d6 ( ) #c0ffc0 ( ) #c0fff0
( ) #c0f0ff ( ) #d0c0ff [✓ #ffc0ff] ( ) #ffc0d0 ( ) #fff0c0 ( ) #c0c0c0

Tags:
- [ ] Add the following new tag name to this check-in:
- [ ] Cancel tag **VER_1.0**
- [ ] Cancel special tag **bgcolor**
- [ ] Cancel special tag **branch**
- [ ] Cancel special tag **date**

Figure 59: Setting Timeline color

Under the **Background Color** section I have checked **Propagate color to descendants** so future checkins will have the same color.

4.4.4 Check out the branches

Now the the repository is branched we can check out the two sets of code into different directories. We create jsonp1 and jsonp2 and proceed to open the different branches into them.

```
[Pandora-2:jschimpf/Public/jsonp1] jim% fossil open ../FOSSIL/jsonp.fossil VER_1.0
docs/qdj.lyx
docs/qdj.pdf
json-src/qdj.c
json-src/qdj.h
json-src/qdj_token.c
json-src/qdj_token.h
json-src/qdj_util.c
main.c
makefile
obj/test.txt
test.txt
project-name: JASONP
repository:   /Users/jschimpf/Public/FOSSIL/jsonp.fossil
local-root:   /Users/jschimpf/Public/jsonp1/
project-code: eb6084c8ab115cf2b28a129c7183731002c6143a
server-code:  90c80f1a2da7360dae230ccec65ff82fe2eb160d
checkout:     65e1f48633d691a5ea738cd51ccbf9a581dfb3c7 2010-06-13 10:13:55 UTC
parent:       462156b283b694af0b99c9b446b64d3f77436fbb 2010-06-09 14:16:42 UTC
tags:         VER_1.0
```

Figure 60: Check out VER_1.0

Checking out VER_2.0 in the same way

```
[Pandora-2:jschimpf/Public/jsonp2] jim% fossil open ../FOSSIL/jsonp.fossil VER_2.0
docs/qdj.lyx
docs/qdj.pdf
json-src/qdj.c
json-src/qdj.h
json-src/qdj_token.c
json-src/qdj_token.h
json-src/qdj_util.c
main.c
makefile
obj/test.txt
test.txt
project-name: JASONP
repository:   /Users/jschimpf/Public/FOSSIL/jsonp.fossil
local-root:   /Users/jschimpf/Public/jsonp2/
project-code: eb6084c8ab115cf2b28a129c7183731002c6143a
server-code:  90c80f1a2da7360dae230ccec65ff82fe2eb160d
checkout:     a1737916ec2df696a0f3a7e36edf9ba4370e48a7 2010-06-13 10:14:26 UTC
parent:       462156b283b694af0b99c9b446b64d3f77436fbb 2010-06-09 14:16:42 UTC
tags:         VER_2.0
```

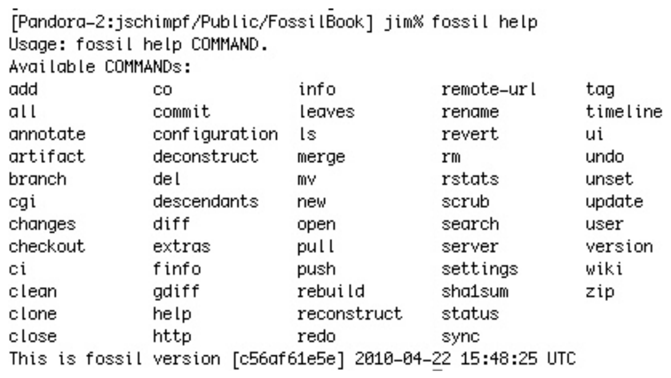
Figure 61: VER_2.0 checkout

Notice on both of these the tags show which branch we are attached to.

4.4.5 Correcting errors (in both)

After doing this work I found that the main.c file had a warning about an unused variable. I wanted to correct this in both branches. At this point all the files in both branches are the same so correcting the file in either branch and copying it to the other is possible. I put in a ticket for the change and edit main.c. I copy it to both checkouts for the both branches and then check both in.

Now the timeline looks like this:



```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help
Usage: fossil help COMMAND.
Available COMMANDs:
add           co            info          remote-url    tag
all           commit        leaves        rename        timeline
annotate      configuration ls            revert        ui
artifact      deconstruct   merge         rm            undo
branch        del           mv            rstats        unset
cgi           descendants   new           scrub         update
changes       diff          open          search        user
checkout      extras        pull          server        version
ci            finfo         push          settings      wiki
clean         gdiff         rebuild       sha1sum       zip
clone         help          reconstruct   status
close         http          redo          sync
This is fossil version [c56af61e5e] 2010-04-22 15:48:25 UTC
```

Figure 63: Correcting both branches

```
[Pandora-2:jschimpf/Public/jsonp1] jim% fossil commit -m "[2795e6c74d] Fix unused variable"
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        2116         46          0          0
Send:             365          6          0          0
Received:        3601         51          5          0
Total network traffic: 805 bytes sent, 3464 bytes received
New_Version: 3b902585d0e8849399286139d27676c5a349de7b
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            3034         50          0          2
Received:        2208         48          0          0
Total network traffic: 1848 bytes sent, 1444 bytes received
[Pandora-2:jschimpf/Public/jsonp1] jim% cd ..
[Pandora-2:/Users/jschimpf/Public] jim% cd jsonp2
[Pandora-2:jschimpf/Public/jsonp2] jim% cp ../jsonp1/main.c .
[Pandora-2:jschimpf/Public/jsonp2] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/jsonp.fossil
local-root:   /Users/jschimpf/Public/jsonp2/
server-code:  90c80f1a2da7360dae230ccec65ff82fe2eb160d
checkout:     a1737916ec2df696a0f3a7e36edf9ba4370e48a7 2010-06-13 10:14:26 UTC
parent:       462156b283b694af0b99c9b446b64d3f77436fbb 2010-06-09 14:16:42 UTC
tags:         VER_2.0
EDITED     main.c
[Pandora-2:jschimpf/Public/jsonp2] jim% fossil commit -m "[2795e6c74d] Fix unused variable"
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:             130          1          0          0
Received:        2392         52          0          0
Send:             318          5          0          0
Received:        3320         56          4          0
Total network traffic: 781 bytes sent, 3508 bytes received
New_Version: 762a31854d708080678598c8d4ce28465cbee8c5
Autosync:  http://jim@pandora.dyn-o-saur.com:8080/cgi-bin/jsonp.cgi
                Bytes      Cards  Artifacts     Deltas
Send:            3253         55          0          2
Received:        2438         53          0          0
Total network traffic: 1972 bytes sent, 1573 bytes received
[Pandora-2:jschimpf/Public/jsonp2] jim%
```

Figure 62: Fixing both branches

4.4.6 Commands used

- **fossil branch** Used to generate a branch of the repository. The command can optionally color the branch in the display.

5 Fossil Commands

5.1 Introduction

This section will go through the various Fossil command line commands. This will be divided into sections, the first will detail the must know commands. These are the ones you will be using all the time and will probably have memorized in short order. The other commands will be divided into Maintenance, Advanced, and Miscellaneous. These you will probably be checking as reference before use.

The most important command is **help**. You can always type **fossil help** at the command line and it will list out all the commands it has. Then typing f**ossil help <command>** will print out the detailed information on that command. You always have that as your reference. This section of the book will try to supplement the built in help with some examples and further explanation of what a command does. All of the commands will be placed in the index for easy searching

NOTE: Fossil is a moving target, commands might be added and others removed future versions. Type **fossil help** on your version to get the latest list. The following applies to the fossil I used when I wrote this and your version might be different. This chapter is based on Fossil version 1.36.

5.2 Basic Commands

5.2.1 help

This command is used to dump the current command set and version of Fossil. It can also be used in the form **fossil help <command>** to get further information on any command.

```
Usage: fossil help COMMAND
Common COMMANDs:  (use "fossil help -a|--all" for a complete list)
add          cat          finfo        mv           rm           ui
addremove    changes      fusefs       open         rss          undo
all          clean        gdiff        praise       settings     unpublished
amend        clone        help         publish      sqlite3      unversioned
annotate     commit       import       pull         stash        update
bisect       delete       info         push         status       version
blame        diff         init         rebuild      sync
branch       export       ls           remote-url   tag
bundle       extras       merge        revert       timeline
This is fossil version 1.36 [c24373934d] 2016-10-24 14:59:33 UTC
```

Figure 64: Help run

Using it to get further information about a particular command

```
505 ~> help help
help: help [-s] [pattern ...]
    Display helpful information about builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise a list of the builtins is printed.  The -s option
    restricts the output for each builtin command matching PATTERN to
    a short usage synopsis.
506 ~>
```

Figure 65: Help detail

5.2.2 add

The add command is used to add files into a repository. It is recursive and will pull in all files in subdirectories of the current. Fossil will not overwrite any of the files already present in the repository so it is safe to add all the files at any time. Only new files will be added.

```
504 ~> fossil help add
Usage: fossil add ?OPTIONS? FILE1 ?FILE2 ...?

Make arrangements to add one or more files or directories to the
current checkout at the next commit.

When adding files or directories recursively, filenames that begin
with "." are excluded by default.  To include such files, add
the "--dotfiles" option to the command-line.

The --ignore and --clean options are comma-separate lists of glob patterns
for files to be excluded.  Example:  '*.o,*.obj,*.exe'  If the --ignore
option does not appear on the command line then the "ignore-glob" setting
is used.  If the --clean option does not appear on the command line then
the "clean-glob" setting is used.

If files are attempted to be added explicitly on the command line which
match "ignore-glob", a confirmation is asked first. This can be prevented
using the -f|--force option.

The --case-sensitive option determines whether or not filenames should
be treated case sensitive or not. If the option is not given, the default
depends on the global setting, or the operating system default, if not set.

Options:

   --case-sensitive <BOOL> Override the case-sensitive setting.
   --dotfiles              include files beginning with a dot (".")
   -f|--force              Add files without prompting
   --ignore <CSG>          Ignore files matching patterns from the
                           comma separated list of glob patterns.
   --clean <CSG>           Also ignore files matching patterns from
                           the comma separated list of glob patterns.

See also: addremove, rm
```

Figure 66: add detail

Typing:

```
fossil add .
```

will add all files in the current directory and subdirectories.

Note none of these files are put in the repository until a commit is done.

5.2.3 addremove

The addremove command command is used when a lot of changes have been made to the checked out directory and it looks like Swiss cheese with a lot of adds and missing files. It allows you to clean up the mess with one simple command. **Important** Before using it make sure you remove any extra files like obj files or intermediate files from building that you don't want in the repository.

5.2.4 rm or delete or forget

The rm command is used to remove files from the repository. The file is not deleted from the file system but it will be dropped from the repository on the next commit. This file will still be available in earlier versions of the repository but not in later ones.

```
502 ~> fossil help rm
Usage: fossil rm|delete|forget FILE1 ?FILE2 ...?

Remove one or more files or directories from the repository.

The 'rm' and 'delete' commands do NOT normally remove the files from
disk.  They just mark the files as no longer being part of the project.
In other words, future changes to the named files will not be versioned.
However, the default behavior of this command may be overridden via the
command line options listed below and/or the 'mv-rm-files' setting.

The 'forget' command never removes files from disk, even when the command
line options and/or the 'mv-rm-files' setting would otherwise require it
to do so.

WARNING: If the "--hard" option is specified -OR- the "mv-rm-files"
         setting is non-zero, files WILL BE removed from disk as well.
         This does NOT apply to the 'forget' command.

Options:
  --soft                  Skip removing files from the checkout.
                          This supersedes the --hard option.
  --hard                  Remove files from the checkout.
  --case-sensitive <BOOL> Override the case-sensitive setting.
  -n|--dry-run            If given, display instead of run actions.

See also: addremove, add
```

Figure 68: rm detail

You can delete groups of files by using wild-cards in their names. Thus if I had a group of files like com_tr.c, com_rx.c and com_mgmt.c I could remove them all with:

```
fossil rm com_*.c
```

By running a "fossil status" you can see what files will be deleted on the next commit.

```
501 ~> fossil help addremove
Usage: fossil addremove ?OPTIONS?

Do all necessary "add" and "rm" commands to synchronize the repository
with the content of the working checkout:

 *  All files in the checkout but not in the repository (that is,
    all files displayed using the "extras" command) are added as
    if by the "add" command.

 *  All files in the repository but missing from the checkout (that is,
    all files that show as MISSING with the "status" command) are
    removed as if by the "rm" command.

The command does not "commit".  You must run the "commit" separately
as a separate step.

Files and directories whose names begin with "." are ignored unless
the --dotfiles option is used.

The --ignore option overrides the "ignore-glob" setting, as do the
--case-sensitive option with the "case-sensitive" setting and the
--clean option with the "clean-glob" setting. See the documentation
on the "settings" command for further information.

The -n|--dry-run option shows what would happen without actually doing
anything.

This command can be used to track third party software.

Options:
  --case-sensitive <BOOL> Override the case-sensitive setting.
  --dotfiles              Include files beginning with a dot (".")
  --ignore <CSG>          Ignore files matching patterns from the
                          comma separated list of glob patterns.
  --clean <CSG>           Also ignore files matching patterns from
                          the comma separated list of glob patterns.
  -n|--dry-run            If given, display instead of run actions.

See also: add, rm
```

Figure 67: addremove detail

5.2.5 rename or mv

This command is used to rename a file in the repository. This does not rename files on disk so is usually used after you have renamed files on the disk then want to change this in the repository.

```
513 ~> fossil help rename
Usage: fossil mv|rename OLDNAME NEWNAME
   or: fossil mv|rename OLDNAME... DIR

Move or rename one or more files or directories within the repository tree.
You can either rename a file or directory or move it to another subdirectory.

The 'mv' command does NOT normally rename or move the files on disk.
This command merely records the fact that file names have changed so
that appropriate notations can be made at the next commit/check-in.
However, the default behavior of this command may be overridden via
command line options listed below and/or the 'mv-rm-files' setting.

The 'rename' command never renames or moves files on disk, even when the
command line options and/or the 'mv-rm-files' setting would otherwise
require it to do so.

WARNING: If the "--hard" option is specified -OR- the "mv-rm-files"
         setting is non-zero, files WILL BE renamed or moved on disk
         as well.  This does NOT apply to the 'rename' command.

Options:
  --soft                  Skip moving files within the checkout.
                          This supersedes the --hard option.
  --hard                  Move files within the checkout.
  --case-sensitive <BOOL> Override the case-sensitive setting.
  -n|--dry-run            If given, display instead of run actions.

See also: changes, status
```

Figure 69: rename detail

Just like add or rm you can use wild cards in the names and rename groups of files. Like them “fossil status” will show you the current state.

5.2.6 status

The status command is used to show you the current state of your files relative to the repository. It will show you files added, deleted, and changed. This is only the condition of files that are already in the repository or under control of fossil. It also shows from where in the timeline you are checked out and where your repository is kept.

```
515 ~> fossil help status
Usage: fossil status ?OPTIONS?

Report on the status of the current checkout.

Pathnames are displayed according to the "relative-paths" setting,
unless overridden by the --abs-paths or --rel-paths options.

Options:

   --abs-paths       Display absolute pathnames.
   --rel-paths       Display pathnames relative to the current working
                     directory.
   --sha1sum         Verify file status using SHA1 hashing rather
                     than relying on file mtimes.

See also: changes, extras, ls
```

Figure 70: status run

The listing above shows where my cloned copy of the repository is kept, where I am working, and the tags show me that I am checked out of the trunk. Finally it shows the status of the files: I am working on two of them.

5.2.7 changes

This lists the changed files like status but does not show the other information that status does.

5.2.8 extra

The extra command is used to find files you have added to your working directory but are not yet under Fossil control. This is important because if you move your working directory or others attempt to use the repository they won't have these files.

```
516 ~> fossil help changes
Usage: fossil changes ?OPTIONS?

Report on the edit status of all files in the current checkout.

Pathnames are displayed according to the "relative-paths" setting,
unless overridden by the --abs-paths or --rel-paths options.

Options:
   --abs-paths       Display absolute pathnames.
   --rel-paths       Display pathnames relative to the current working
                     directory.
   --sha1sum         Verify file status using SHA1 hashing rather
                     than relying on file mtimes.
   --header          Identify the repository if there are changes
   -v|--verbose      Say "(none)" if there are no changes

See also: extras, ls, status
```

Figure 71: changes details

```
517 ~> fossil help extra
Usage: fossil extras ?OPTIONS? ?PATH1 ...?

Print a list of all files in the source tree that are not part of the
current checkout. See also the "clean" command. If paths are specified,
only files in the given directories will be listed.

Files and subdirectories whose names begin with "." are normally
ignored but can be included by adding the --dotfiles option.

Files whose names match any of the glob patterns in the "ignore-glob"
setting are ignored. This setting can be overridden by the --ignore
option, whose CSG argument is a comma-separated list of glob patterns.

Pathnames are displayed according to the "relative-paths" setting,
unless overridden by the --abs-paths or --rel-paths options.

Options:
   --abs-paths      Display absolute pathnames.
   --case-sensitive <BOOL> override case-sensitive setting
   --dotfiles       include files beginning with a dot (".")
   --header         Identify the repository if there are extras
   --ignore <CSG>   ignore files matching patterns from the argument
   --rel-paths      Display pathnames relative to the current working
                    directory.

See also: changes, clean, status
```

Figure 72: extra details

The –dotfiles option shows you any files starting with "." that are not under Fossil control. This would be important if you need those files in your repository. The last option –ignore allows you to ignore certain files you know don't belong in the repository. In my case there is a file called fossilbook.lyx~ that is a LYX backup file that I do not want, as it is temporary. So I can say

```
fossil extra --ignore *.lyx~
```

and only get:

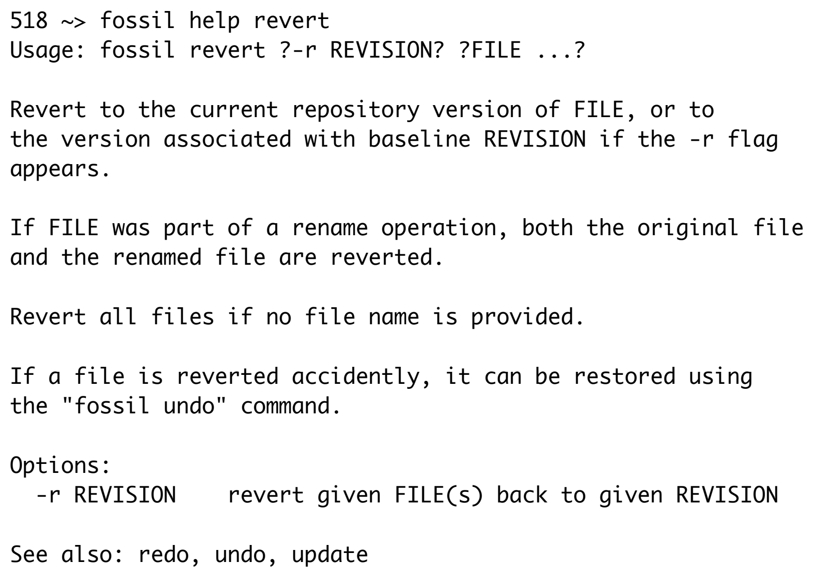
```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra --ignore *.lyx~
Images/Commands/help1.epsf
```

instead of:

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra
Images/Commands/help1.epsf
fossilbook.lyx~
```

5.2.9 revert

The revert command is used to take a file back to the value in the repository. This is useful when you make a error in editing or other mistake.



```
518 ~> fossil help revert
Usage: fossil revert ?-r REVISION? ?FILE ...?

Revert to the current repository version of FILE, or to
the version associated with baseline REVISION if the -r flag
appears.

If FILE was part of a rename operation, both the original file
and the renamed file are reverted.

Revert all files if no file name is provided.

If a file is reverted accidently, it can be restored using
the "fossil undo" command.

Options:
  -r REVISION    revert given FILE(s) back to given REVISION

See also: redo, undo, update
```

Figure 73: revert details

With no parameters it will revert the file to the current revision, see Figure 70 on page 59. The -r option allows you to pick any revision from the time line.

5.2.10 update

The update option will update a file or files to match the repository. With multiple users it should be done before you start working on any files. This ensures you have the latest version of all the files.

```
519 ~> fossil help update
Usage: fossil update ?OPTIONS? ?VERSION? ?FILES...?

Change the version of the current checkout to VERSION.  Any
uncommitted changes are retained and applied to the new checkout.

The VERSION argument can be a specific version or tag or branch
name.  If the VERSION argument is omitted, then the leaf of the
subtree that begins at the current version is used, if there is
only a single leaf.  VERSION can also be "current" to select the
leaf of the current version or "latest" to select the most recent
check-in.

If one or more FILES are listed after the VERSION then only the
named files are candidates to be updated, and any updates to them
will be treated as edits to the current version. Using a directory
name for one of the FILES arguments is the same as using every
subdirectory and file beneath that directory.

If FILES is omitted, all files in the current checkout are subject
to being updated and the version of the current checkout is changed
to VERSION. Any uncommitted changes are retained and applied to the
new checkout.

The -n or --dry-run option causes this command to do a "dry run".
It prints out what would have happened but does not actually make
any changes to the current checkout or the repository.

The -v or --verbose option prints status information about
unchanged files in addition to those file that actually do change.

Options:
  --case-sensitive <BOOL> override case-sensitive setting
  --debug          print debug information on stdout
  --latest         acceptable in place of VERSION, update to latest version
  --force-missing  force update if missing content after sync
  -n|--dry-run     If given, display instead of run actions
  -v|--verbose     print status information about all files
  -W|--width <num> Width of lines (default is to auto-detect). Must be >20
                   or 0 (= no limit, resulting in a single line per entry).

See also: revert
```

Figure 74: update details

Update has a number of options, first you can tie the update to a particular version, if not picked then it just uses the latest. Second it can work on a single files or many files at once. That is you could say

```
fossil update *.c
```

and it would update all C files.

Since this is a rather large set of changes it has a special “dry run” mode. If you add -n on the command it will just print out what will be done but not do it. This is very useful to do this trial if you are unsure what might happen. The -v command (which can be used with -n or alone) prints out the action for each file even if it does nothing.

5.2.11 checkout or co

This command is similar to update.

```
520 ~> fossil help checkout
Usage: fossil checkout ?VERSION | --latest? ?OPTIONS?
   or: fossil co ?VERSION | --latest? ?OPTIONS?

Check out a version specified on the command-line.  This command
will abort if there are edited files in the current checkout unless
the --force option appears on the command-line.  The --keep option
leaves files on disk unchanged, except the manifest and manifest.uuid
files.

The --latest flag can be used in place of VERSION to checkout the
latest version in the repository.

Options:
   --force          Ignore edited files in the current checkout
   --keep           Only update the manifest and manifest.uuid files
   --force-missing  Force checkout even if content is missing

See also: update
```

Figure 75: checkout or co details

5.2.12 undo

This is used to undo the last update, merge, or revert operation.

```
521 ~> fossil help undo
Usage: fossil undo ?OPTIONS? ?FILENAME...?
   or: fossil redo ?OPTIONS? ?FILENAME...?

Undo the changes to the working checkout caused by the most recent
of the following operations:

   (1) fossil update             (5) fossil stash apply
   (2) fossil merge              (6) fossil stash drop
   (3) fossil revert             (7) fossil stash goto
   (4) fossil stash pop

The "fossil clean" operation can also be undone; however, this is
currently limited to files that are less than 10MiB in size.

If FILENAME is specified then restore the content of the named
file(s) but otherwise leave the update or merge or revert in effect.
The redo command undoes the effect of the most recent undo.

If the -n|--dry-run option is present, no changes are made and instead
the undo or redo command explains what actions the undo or redo would
have done had the -n|--dry-run been omitted.

A single level of undo/redo is supported.  The undo/redo stack
is cleared by the commit and checkout commands.

Options:
  -n|--dry-run   do not make changes but show what would be done

See also: commit, status
```

Figure 76: undo details

It acts on a single file or files if specified, otherwise if no file given, it undoes all of the last changes.

5.2.13 diff

The diff command is used to produce a text listing of the difference of a file in the working directory and that same file in the repository. If you don't specify a file it will show the differences between all the changed files in the working directory vs the repository. If you use the –from and –to options you can specify which versions to check and to compare between two different versions in the repository. Not using the –to means compare with the working directory.

If you have configured an external diff program it will be used unless you use the -i option which uses the diff built into Fossil.

```
522 ~> fossil help diff
Usage: fossil diff|gdiff ?OPTIONS? ?FILE1? ?FILE2 ...?

Show the difference between the current version of each of the FILEs
specified (as they exist on disk) and that same file as it was checked
out.  Or if the FILE arguments are omitted, show the unsaved changes
currently in the working check-out.

If the "--from VERSION" or "-r VERSION" option is used it specifies
the source check-in for the diff operation.  If not specified, the
source check-in is the base check-in for the current check-out.

If the "--to VERSION" option appears, it specifies the check-in from
which the second version of the file or files is taken.  If there is
no "--to" option then the (possibly edited) files in the current check-out
are used.

The "--checkin VERSION" option shows the changes made by
check-in VERSION relative to its primary parent.

The "-i" command-line option forces the use of the internal diff logic
rather than any external diff program that might be configured using
the "setting" command.  If no external diff program is configured, then
the "-i" option is a no-op.  The "-i" option converts "gdiff" into "diff".

The "-N" or "--new-file" option causes the complete text of added or
deleted files to be displayed.

The "--diff-binary" option enables or disables the inclusion of binary files
when using an external diff program.

The "--binary" option causes files matching the glob PATTERN to be treated
as binary when considering if they should be used with external diff program.
This option overrides the "binary-glob" setting.

Options:
  --binary PATTERN            Treat files that match the glob PATTERN as binary
  --branch BRANCH             Show diff of all changes on BRANCH
  --brief                     Show filenames only
  --checkin VERSION           Show diff of all changes in VERSION
  --context|-c N              Use N lines of context
  --diff-binary BOOL          Include binary files when using external commands
  --exec-abs-paths            Force absolute path names with external commands.
  --exec-rel-paths            Force relative path names with external commands.
  --from|-r VERSION           Select VERSION as source for the diff
  --internal|-i               Use internal diff logic
  --side-by-side|-y           Side-by-side diff
  --strip-trailing-cr         Strip trailing CR
  --tk                        Launch a Tcl/Tk GUI for display
  --to VERSION                Select VERSION as target for the diff
  --undo                      Diff against the "undo" buffer
  --unified                   Unified diff
  -v|--verbose                Output complete text of added or deleted files
  -w|--ignore-all-space       Ignore white space when comparing lines
  -W|--width <num>            Width of lines in side-by-side diff
  -Z|--ignore-trailing-space Ignore changes to end-of-line whitespace
```

Figure 77: diff details

5.2.14 gdiff

This is the same as the diff command but uses (if configured) a graphical diff program you have on your system. See the settings command for details on how to set the graphical diff program.

5.2.15 ui

The ui command is used to start Fossil in a local webserver. The –port option is used to specify the port it uses, by default it uses 8080. It should automatically start the system's web browser and it will come up with the repository web page. If run within a working directory it will bring up the web page for that repository. If run outside the working directory you can specify the repository on the command line.

```
523 ~> fossil help gdiff
Usage: fossil diff|gdiff ?OPTIONS? ?FILE1? ?FILE2 ...?

Show the difference between the current version of each of the FILEs
specified (as they exist on disk) and that same file as it was checked
out.  Or if the FILE arguments are omitted, show the unsaved changes
currently in the working check-out.

If the "--from VERSION" or "-r VERSION" option is used it specifies
the source check-in for the diff operation.  If not specified, the
source check-in is the base check-in for the current check-out.

If the "--to VERSION" option appears, it specifies the check-in from
which the second version of the file or files is taken.  If there is
no "--to" option then the (possibly edited) files in the current check-out
are used.

The "--checkin VERSION" option shows the changes made by
check-in VERSION relative to its primary parent.

The "-i" command-line option forces the use of the internal diff logic
rather than any external diff program that might be configured using
the "setting" command.  If no external diff program is configured, then
the "-i" option is a no-op.  The "-i" option converts "gdiff" into "diff".

The "-N" or "--new-file" option causes the complete text of added or
deleted files to be displayed.

The "--diff-binary" option enables or disables the inclusion of binary files
when using an external diff program.

The "--binary" option causes files matching the glob PATTERN to be treated
as binary when considering if they should be used with external diff program.
This option overrides the "binary-glob" setting.

Options:
  --binary PATTERN           Treat files that match the glob PATTERN as binary
  --branch BRANCH            Show diff of all changes on BRANCH
  --brief                    Show filenames only
  --checkin VERSION          Show diff of all changes in VERSION
  --context|-c N             Use N lines of context
  --diff-binary BOOL         Include binary files when using external commands
  --exec-abs-paths           Force absolute path names with external commands.
  --exec-rel-paths           Force relative path names with external commands.
  --from|-r VERSION          Select VERSION as source for the diff
  --internal|-i              Use internal diff logic
  --side-by-side|-y          Side-by-side diff
  --strip-trailing-cr        Strip trailing CR
  --tk                       Launch a Tcl/Tk GUI for display
  --to VERSION               Select VERSION as target for the diff
  --undo                     Diff against the "undo" buffer
  --unified                  Unified diff
  -v|--verbose               Output complete text of added or deleted files
  -w|--ignore-all-space      Ignore white space when comparing lines
  -W|--width <num>           Width of lines in side-by-side diff
  -Z|--ignore-trailing-space Ignore changes to end-of-line whitespace
```

Figure 78: gdiff details

```
524 ~> fossil help ui
Usage: fossil server ?OPTIONS? ?REPOSITORY?
   or: fossil ui ?OPTIONS? ?REPOSITORY?

Open a socket and begin listening and responding to HTTP requests on
TCP port 8080, or on any other TCP port defined by the -P or
--port option.  The optional argument is the name of the repository.
The repository argument may be omitted if the working directory is
within an open checkout.

The "ui" command automatically starts a web browser after initializing
the web server.  The "ui" command also binds to 127.0.0.1 and so will
only process HTTP traffic from the local machine.

The REPOSITORY can be a directory (aka folder) that contains one or
more repositories with names ending in ".fossil".  In this case, a
prefix of the URL pathname is used to search the directory for an
appropriate repository.  To thwart mischief, the pathname in the URL must
contain only alphanumerics, "_", "/", "-", and ".", and no "-" may
occur after "/", and every "." must be surrounded on both sides by
alphanumerics.  Any pathname that does not satisfy these constraints
results in a 404 error.  Files in REPOSITORY that match the comma-separated
list of glob patterns given by --files and that have known suffixes
such as ".txt" or ".html" or ".jpeg" and do not match the pattern
"*.fossil*" will be served as static content.  With the "ui" command,
the REPOSITORY can only be a directory if the --notfound option is
also present.

By default, the "ui" command provides full administrative access without
having to log in.  This can be disabled by turning off the "localauth"
setting.  Automatic login for the "server" command is available if the
--localauth option is present and the "localauth" setting is off and the
connection is from localhost.  The "ui" command also enables --repolist
by default.

Options:
  --baseurl URL       Use URL as the base (useful for reverse proxies)
  --create            Create a new REPOSITORY if it does not already exist
  --page PAGE         Start "ui" on PAGE.  ex: --page "timeline?y=ci"
  --files GLOBLIST    Comma-separated list of glob patterns for static files
  --localauth         enable automatic login for requests from localhost
  --localhost         listen on 127.0.0.1 only (always true for "ui")
  --https             signal a request coming in via https
  --nojail            Drop root privileges but do not enter the chroot jail
  --nossl             signal that no SSL connections are available
  --notfound URL      Redirect
  -P|--port TCPPORT   listen to request on port TCPPORT
  --th-trace          trace TH1 execution (for debugging purposes)
  --repolist          If REPOSITORY is dir, URL "/" lists repos.
  --scgi              Accept SCGI rather than HTTP
  --skin LABEL        Use override skin LABEL

See also: cgi, http, winsrv
```

Figure 79: ui details

5.2.16 server

This is a more powerful version of the ui command. This allows you to have multiple repositories supported by a single running Fossil webserver. This way you start the server and instead of a particular repository you specify a directory where a number of repositories reside (all having the extension .fossil) then you can open and use any of them.

```
525 ~> fossil help server
Usage: fossil server ?OPTIONS? ?REPOSITORY?
   or: fossil ui ?OPTIONS? ?REPOSITORY?

Open a socket and begin listening and responding to HTTP requests on
TCP port 8080, or on any other TCP port defined by the -P or
--port option.  The optional argument is the name of the repository.
The repository argument may be omitted if the working directory is
within an open checkout.

The "ui" command automatically starts a web browser after initializing
the web server.  The "ui" command also binds to 127.0.0.1 and so will
only process HTTP traffic from the local machine.

The REPOSITORY can be a directory (aka folder) that contains one or
more repositories with names ending in ".fossil".  In this case, a
prefix of the URL pathname is used to search the directory for an
appropriate repository.  To thwart mischief, the pathname in the URL must
contain only alphanumerics, "_", "/", "-", and ".", and no "-" may
occur after "/", and every "." must be surrounded on both sides by
alphanumerics.  Any pathname that does not satisfy these constraints
results in a 404 error.  Files in REPOSITORY that match the comma-separated
list of glob patterns given by --files and that have known suffixes
such as ".txt" or ".html" or ".jpeg" and do not match the pattern
"*.fossil*" will be served as static content.  With the "ui" command,
the REPOSITORY can only be a directory if the --notfound option is
also present.

By default, the "ui" command provides full administrative access without
having to log in.  This can be disabled by turning off the "localauth"
setting.  Automatic login for the "server" command is available if the
--localauth option is present and the "localauth" setting is off and the
connection is from localhost.  The "ui" command also enables --repolist
by default.

Options:
  --baseurl URL       Use URL as the base (useful for reverse proxies)
  --create            Create a new REPOSITORY if it does not already exist
  --page PAGE         Start "ui" on PAGE.  ex: --page "timeline?y=ci"
  --files GLOBLIST    Comma-separated list of glob patterns for static files
  --localauth         enable automatic login for requests from localhost
  --localhost         listen on 127.0.0.1 only (always true for "ui")
  --https             signal a request coming in via https
  --nojail            Drop root privileges but do not enter the chroot jail
  --nossl             signal that no SSL connections are available
  --notfound URL      Redirect
  -P|--port TCPPORT   listen to request on port TCPPORT
  --th-trace          trace TH1 execution (for debugging purposes)
  --repolist          If REPOSITORY is dir, URL "/" lists repos.
  --scgi              Accept SCGI rather than HTTP
  --skin LABEL        Use override skin LABEL

See also: cgi, http, winsrv
```

Figure 80: server detail

5.2.17 commit or ci

This is the command used to put the current changes in the working directory into the repository, giving this a new version and updating the timeline.

```
526 ~> fossil help commit
Usage: fossil commit ?OPTIONS? ?FILE...?

Create a new version containing all of the changes in the current
checkout.  You will be prompted to enter a check-in comment unless
the comment has been specified on the command-line using "-m" or a
file containing the comment using -M.  The editor defined in the
"editor" fossil option (see fossil help set) will be used, or from
the "VISUAL" or "EDITOR" environment variables (in that order) if
no editor is set.

All files that have changed will be committed unless some subset of
files is specified on the command line.

The --branch option followed by a branch name causes the new
check-in to be placed in a newly-created branch with the name
passed to the --branch option.

Use the --branchcolor option followed by a color name (ex:
'#ffc0c0') to specify the background color of entries in the new
branch when shown in the web timeline interface.  The use of
the --branchcolor option is not recommended.  Instead, let Fossil
choose the branch color automatically.

The --bgcolor option works like --branchcolor but only sets the
background color for a single check-in.  Subsequent check-ins revert
to the default color.

A check-in is not permitted to fork unless the --allow-fork option
appears.  An empty check-in (i.e. with nothing changed) is not
allowed unless the --allow-empty option appears.  A check-in may not
be older than its ancestor unless the --allow-older option appears.
If any of files in the check-in appear to contain unresolved merge
conflicts, the check-in will not be allowed unless the
--allow-conflict option is present.  In addition, the entire
check-in process may be aborted if a file contains content that
appears to be binary, Unicode text, or text with CR/NL line endings
unless the interactive user chooses to proceed.  If there is no
interactive user or these warnings should be skipped for some other
reason, the --no-warnings option may be used.  A check-in is not
allowed against a closed leaf.

If a commit message is blank, you will be prompted:
("continue (y/N)?") to confirm you really want to commit with a
blank commit message.  The default value is "N", do not commit.

The --private option creates a private check-in that is never synced.
Children of private check-ins are automatically private.

The --tag option applies the symbolic tag name to the check-in.

The --sha1sum option detects edited files by computing each file's
SHA1 hash rather than just checking for changes to its size or mtime.

Options:
   --allow-conflict          allow unresolved merge conflicts
   --allow-empty             allow a commit with no changes
   --allow-fork              allow the commit to fork
   --allow-older             allow a commit older than its ancestor
   --baseline                use a baseline manifest in the commit process
   --bgcolor COLOR           apply COLOR to this one check-in only
   --branch NEW-BRANCH-NAME  check in to this new branch
```

It's a very good idea to always put a comment (-comment or -m) text on any commit. This way you get documentation in the timeline.

5.3 Maintenance commands

These commands you will probably use less often since the actions they perform are not needed in normal operation. You will have to use them and referring here or to **fossil help** will probably be required before use. Some of them like new or clone are only needed when you start a repository. Others like rebuild or reconstruct are only needed to fix or update a repository.

5.3.1 new

This command is used to create a new repository.

```
528 ~> fossil help new
Usage: fossil new ?OPTIONS? FILENAME
   or: fossil init ?OPTIONS? FILENAME

Create a repository for a new project in the file named FILENAME.
This command is distinct from "clone".  The "clone" command makes
a copy of an existing project.  This command starts a new project.

By default, your current login name is used to create the default
admin user. This can be overridden using the -A|--admin-user
parameter.

By default, all settings will be initialized to their default values.
This can be overridden using the --template parameter to specify a
repository file from which to copy the initial settings.  When a template
repository is used, almost all of the settings accessible from the setup
page, either directly or indirectly, will be copied.  Normal users and
their associated permissions will not be copied; however, the system
default users "anonymous", "nobody", "reader", "developer", and their
associated permissions will be copied.

Options:
   --template      FILE      copy settings from repository file
   --admin-user|-A USERNAME  select given USERNAME as admin user
   --date-override DATETIME  use DATETIME as time of the initial check-in

DATETIME may be "now" or "YYYY-MM-DDTHH:MM:SS.SSS". If in
year-month-day form, it may be truncated, the "T" may be replaced by
a space, and it may also name a timezone offset from UTC as "-HH:MM"
(westward) or "+HH:MM" (eastward). Either no timezone suffix or "Z"
means UTC.

See also: clone
```

Figure 82: new details

The file name specifies the new repository name. The options provided allow you to specify the admin user name if you want it to be different than your current login and the starting date if you want it to be different than now.

5.3.2 clone

The clone command is used to create your own local version of the master repository. If you are supporting multiple users via a network accessible version of the original repository (see Section 3.2.1 on page 28), then this command will copy that repository to your machine. Also it will make a link between your copy and the master, so that changes made in your copy will be propagated to the master.

```
529 ~> fossil help clone
Usage: fossil clone ?OPTIONS? URI FILENAME

Make a clone of a repository specified by URI in the local
file named FILENAME.

URI may be one of the following form: ([...] mean optional)
  HTTP/HTTPS protocol:
    http[s]://[userid[:password]@]host[:port][/path]

  SSH protocol:
    ssh://[userid@]host[:port]/path/to/repo.fossil\
    [?fossil=path/to/fossil.exe]

  Filesystem:
    [file://]path/to/repo.fossil

Note 1: For ssh and filesystem, path must have an extra leading
        '/' to use an absolute path.

Note 2: Use %HH escapes for special characters in the userid and
        password.  For example "%40" in place of "@", "%2f" in place
        of "/", and "%3a" in place of ":".

By default, your current login name is used to create the default
admin user. This can be overridden using the -A|--admin-user
parameter.

Options:
   --admin-user|-A USERNAME   Make USERNAME the administrator
   --once                     Don't remember the URI.
   --private                  Also clone private branches
   --ssl-identity FILENAME    Use the SSL identity if requested by the server
   --ssh-command|-c SSH       Use SSH as the "ssh" command
   --httpauth|-B USER:PASS    Add HTTP Basic Authorization to requests
   -u|--unversioned           Also sync unversioned content
   -v|--verbose               Show more statistics in output

See also: init
```

Figure 83: clone details

Just like create you can specify the admin user for this clone with an option. The URL for the master repository is of the form:

```
http://<user>:<password>@domain
```

Where **user** and **password** are for a valid user of the selected repository. It is best to check the path with a browser before doing the clone. Make sure you can reach it, for example the repository for this book is:

```
http://pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi
```

Putting that into a browser should get you the home page for this book. (See Figure 29 on page 30). After you have verified that, then running the clone command should work.

Don't forget (as I always do) to put in the file name for the local repository, (see FILENAME above)

5.3.3 open

The open command is used to copy the files in a repository to a working directory. Doing this allows you to build or modify the product. The command also links this working directory to the repository so commits will go into the repository.

```
530 ~> fossil help open
Usage: fossil open FILENAME ?VERSION? ?OPTIONS?

Open a connection to the local repository in FILENAME.  A checkout
for the repository is created with its root at the working directory.
If VERSION is specified then that version is checked out.  Otherwise
the latest version is checked out.  No files other than "manifest"
and "manifest.uuid" are modified if the --keep option is present.

Options:
  --empty           Initialize checkout as being empty, but still connected
                    with the local repository. If you commit this checkout,
                    it will become a new "initial" commit in the repository.
  --keep            Only modify the manifest and manifest.uuid files
  --nested          Allow opening a repository inside an opened checkout
  --force-missing   Force opening a repository with missing content

See also: close
```

Figure 84: open details

If you have multiple users or have a branched repository then it is probably wise to specify the particular version you want. When you run this it will create all the files and directories in the repository in your work area. In addition the files _FOSSIL_, manifest and manifest.uuid will be created by Fossil.

5.3.4 close

This is the opposite of open, in that it breaks the connection between this working directory and the Fossil repository.

```
530 ~> fossil help open
Usage: fossil open FILENAME ?VERSION? ?OPTIONS?

Open a connection to the local repository in FILENAME.  A checkout
for the repository is created with its root at the working directory.
If VERSION is specified then that version is checked out.  Otherwise
the latest version is checked out.  No files other than "manifest"
and "manifest.uuid" are modified if the --keep option is present.

Options:
  --empty           Initialize checkout as being empty, but still connected
                    with the local repository. If you commit this checkout,
                    it will become a new "initial" commit in the repository.
  --keep            Only modify the manifest and manifest.uuid files
  --nested          Allow opening a repository inside an opened checkout
  --force-missing   Force opening a repository with missing content

See also: close
531 ~> fossil help close
Usage: fossil close ?OPTIONS?

The opposite of "open".  Close the current database connection.
Require a -f or --force flag if there are unsaved changes in the
current check-out or if there is non-empty stash.

Options:
  --force|-f  necessary to close a check out with uncommitted changes

See also: open
```

Figure 85: close details

This is useful if you need to abandon the current working directory. Fossil will not let you do this if there are changes between the current directory and the repository. With the force flag you can explicitly cut the connection even if there are changes.

5.3.5 version

This command is used to show the current version of fossil.

```
532 ~> fossil help version
Usage: fossil version ?-verbose|-v?

Print the source code version number for the fossil executable.
If the verbose option is specified, additional details will
be output about what optional features this binary was compiled
with
```

Figure 86: version details

The above figure shows the help and example of running the command. When you have problems with fossil it is very important to have this version information. You can then inquire of the Fossil news group about this problem and with the version information they can easily tell you if the problem is fixed already or is new.

5.3.6 rebuild

If you update your copy of Fossil you will want to run this command against all the repositories you have. This will automatically update them to the new version of Fossil.

```
533 ~> fossil help rebuild
Usage: fossil rebuild ?REPOSITORY? ?OPTIONS?

Reconstruct the named repository database from the core
records.  Run this command after updating the fossil
executable in a way that changes the database schema.

Options:
  --analyze         Run ANALYZE on the database after rebuilding
  --cluster         Compute clusters for unclustered artifacts
  --compress        Strive to make the database as small as possible
  --compress-only   Skip the rebuilding step. Do --compress only
  --deanalyze       Remove ANALYZE tables from the database
  --force           Force the rebuild to complete even if errors are seen
  --ifneeded        Only do the rebuild if it would change the schema version
  --index           Always add in the full-text search index
  --noverify        Skip the verification of changes to the BLOB table
  --noindex         Always omit the full-text search index
  --pagesize N      Set the database pagesize to N. (512..65536 and power of 2)
  --quiet           Only show output if there are errors
  --randomize       Scan artifacts in a random order
  --stats           Show artifact statistics after rebuilding
  --vacuum          Run VACUUM on the database after rebuilding
  --wal             Set Write-Ahead-Log journalling mode on the database

See also: deconstruct, reconstruct
```

Figure 87: rebuild details

5.3.7 all

This command is actually a modifier and when used before certain commands will run them on all the repositories.

```
534 ~> fossil help all
Usage: fossil all SUBCOMMAND ...

The ~/.fossil file records the location of all repositories for a
user.  This command performs certain operations on all repositories
that can be useful before or after a period of disconnected operation.

On Win32 systems, the file is named "_fossil" and is located in
%LOCALAPPDATA%, %APPDATA% or %HOMEPATH%.

Available operations are:

   cache        Manages the cache used for potentially expensive web
                pages.  Any additional arguments are passed on verbatim
                to the cache command.

   changes      Shows all local checkouts that have uncommitted changes.
                This operation has no additional options.

   clean        Delete all "extra" files in all local checkouts.  Extreme
                caution should be exercised with this command because its
                effects cannot be undone.  Use of the --dry-run option to
                carefully review the local checkouts to be operated upon
                and the --whatif option to carefully review the files to
                be deleted beforehand is highly recommended.  The command
                line options supported by the clean command itself, if any
                are present, are passed along verbatim.

   config       Only the "config pull AREA" command works.

   dbstat       Run the "dbstat" command on all repositories.

   extras       Shows "extra" files from all local checkouts.  The command
                line options supported by the extra command itself, if any
                are present, are passed along verbatim.

   fts-config   Run the "fts-config" command on all repositories.

   info         Run the "info" command on all repositories.

   pull         Run a "pull" operation on all repositories.  Only the
                --verbose option is supported.

   push         Run a "push" on all repositories.  Only the --verbose
                option is supported.

   rebuild      Rebuild on all repositories.  The command line options
                supported by the rebuild command itself, if any are
                present, are passed along verbatim.  The --force and
                --randomize options are not supported.

   sync         Run a "sync" on all repositories.  Only the --verbose
                option is supported.

   setting      Run the "setting", "set", or "unset" commands on all
   set          repositories.  These command are particularly useful in
   unset        conjunction with the "max-loadavg" setting which cannot
                otherwise be set globally.

In addition, the following maintenance operations are supported:

   add          Add all the repositories named to the set of repositories
```

5.3.8 push

This command will push changes in the local repository to the master or remote repository.

```
535 ~> fossil help push
Usage: fossil push ?URL? ?options?

Push all sharable changes from the local repository to a remote repository.
Sharable changes include public check-ins, and wiki, ticket, and tech-note
edits.  Use --private to also push private branches.  Use the
"configuration push" command to push website configuration details.

If URL is not specified, then the URL from the most recent clone, push,
pull, remote-url, or sync command is used.  See "fossil help clone" for
details on the URL formats.

Options:

  -B|--httpauth USER:PASS    Credentials for the simple HTTP auth protocol,
                             if required by the remote website
  --ipv4                     Use only IPv4, not IPv6
  --once                     Do not remember URL for subsequent syncs
  --proxy PROXY              Use the specified HTTP proxy
  --private                  Push private branches too
  -R|--repository REPO       Repository to pull into
  --ssl-identity FILE        Local SSL credentials, if requested by remote
  --ssh-command SSH          Use SSH as the "ssh" command
  -v|--verbose               Additional (debugging) output
  --verily                   Exchange extra information with the remote
                             to ensure no content is overlooked

See also: clone, config push, pull, remote-url, sync
```

Figure 89: push details

5.3.9 pull

This command will copy changes from the remote repository to the local repository. You could then use **update** to apply these changes to checked out files.

```
536 ~> fossil help pull
Usage: fossil pull ?URL? ?options?

Pull all sharable changes from a remote repository into the local repository.
Sharable changes include public check-ins, and wiki, ticket, and tech-note
edits.  Add the --private option to pull private branches.  Use the
"configuration pull" command to pull website configuration details.

If URL is not specified, then the URL from the most recent clone, push,
pull, remote-url, or sync command is used.  See "fossil help clone" for
details on the URL formats.

Options:

  -B|--httpauth USER:PASS    Credentials for the simple HTTP auth protocol,
                             if required by the remote website
  --from-parent-project      Pull content from the parent project
  --ipv4                     Use only IPv4, not IPv6
  --once                     Do not remember URL for subsequent syncs
  --proxy PROXY              Use the specified HTTP proxy
  --private                  Pull private branches too
  -R|--repository REPO       Repository to pull into
  --ssl-identity FILE        Local SSL credentials, if requested by remote
  --ssh-command SSH          Use SSH as the "ssh" command
  -v|--verbose               Additional (debugging) output
  --verily                   Exchange extra information with the remote
                             to ensure no content is overlooked

See also: clone, config pull, push, remote-url, sync
```

Figure 90: pull details

5.3.10 sync

This command is used to sync a remote copy with the original copy of the repository, it does both a push and pull. This can also be used to switch a local repository to a different main repository by specifying the URL of a remote repository. If you want to run the update command with -n where it does a dry run, this does not do a sync first so doing fossil sync then fossil update -n will do that for you.

```
537 ~> fossil help sync
Usage: fossil sync ?URL? ?options?

Synchronize all sharable changes between the local repository and a
remote repository.  Sharable changes include public check-ins and
edits to wiki pages, tickets, and technical notes.

If URL is not specified, then the URL from the most recent clone, push,
pull, remote-url, or sync command is used.  See "fossil help clone" for
details on the URL formats.

Options:

  -B|--httpauth USER:PASS    Credentials for the simple HTTP auth protocol,
                             if required by the remote website
  --ipv4                     Use only IPv4, not IPv6
  --once                     Do not remember URL for subsequent syncs
  --proxy PROXY              Use the specified HTTP proxy
  --private                  Sync private branches too
  -R|--repository REPO       Repository to pull into
  --ssl-identity FILE        Local SSL credentials, if requested by remote
  --ssh-command SSH          Use SSH as the "ssh" command
  -u|--unversioned           Also sync unversioned content
  -v|--verbose               Additional (debugging) output
  --verily                   Exchange extra information with the remote
                             to ensure no content is overlooked

See also: clone, pull, push, remote-url
```

Figure 91: sync details

5.3.11 clean

This call can be used to remove all the “extra” files in a source tree. This is useful if you wish to tidy up a source tree or to do a clean build.

```
538 ~> fossil help clean
Usage: fossil clean ?OPTIONS? ?PATH ...?

Delete all "extra" files in the source tree.  "Extra" files are files
that are not officially part of the checkout.  If one or more PATH
arguments appear, then only the files named, or files contained with
directories named, will be removed.

If the --prompt option is used, prompts are issued to confirm the
permanent removal of each file.  Otherwise, files are backed up to the
undo buffer prior to removal, and prompts are issued only for files
whose removal cannot be undone due to their large size or due to
--disable-undo being used.

The --force option treats all prompts as having been answered yes,
whereas --no-prompt treats them as having been answered no.

Files matching any glob pattern specified by the --clean option are
deleted without prompting, and the removal cannot be undone.

No file that matches glob patterns specified by --ignore or --keep will
ever be deleted.  Files and subdirectories whose names begin with "."
are automatically ignored unless the --dotfiles option is used.

The default values for --clean, --ignore, and --keep are determined by
the (versionable) clean-glob, ignore-glob, and keep-glob settings.

The --verily option ignores the keep-glob and ignore-glob settings and
turns on --force, --emptydirs, --dotfiles, and --disable-undo.  Use the
--verily option when you really want to clean up everything.  Extreme
care should be exercised when using the --verily option.

Options:
   --allckouts      Check for empty directories within any checkouts
                    that may be nested within the current one.  This
                    option should be used with great care because the
                    empty-dirs setting (and other applicable settings)
                    belonging to the other repositories, if any, will
                    not be checked.
   --case-sensitive <BOOL> override case-sensitive setting
   --dirsonly       Only remove empty directories.  No files will
                    be removed.  Using this option will automatically
                    enable the --emptydirs option as well.
   --disable-undo   WARNING: This option disables use of the undo
                    mechanism for this clean operation and should be
                    used with extreme caution.
   --dotfiles       Include files beginning with a dot (".").
   --emptydirs      Remove any empty directories that are not
                    explicitly exempted via the empty-dirs setting
                    or another applicable setting or command line
                    argument.  Matching files, if any, are removed
                    prior to checking for any empty directories;
                    therefore, directories that contain only files
                    that were removed will be removed as well.
   -f|--force       Remove files without prompting.
   -i|--prompt      Prompt before removing each file.  This option
                    implies the --disable-undo option.
   -x|--verily      WARNING: Removes everything that is not a managed
                    file or the repository itself.  This option
                    implies the --force, --emptydirs, --dotfiles, and
                    --disable-undo options.  Furthermore, it completely
                    disregards the keep-glob and ignore-glob settings.
```

5.3.12 branch

This command is used if you want to create or list branches in a repository. Previously we discussed forks (See Section 3.4.3 on page 35); branches are the same idea but under user control. This would be where you have version 1.0 of something but want to branch off version 2.0 to add new features but want to keep a 1.0 branch for maintenance.

```
539 ~> fossil help branch
Usage: fossil branch SUBCOMMAND ... ?OPTIONS?

Run various subcommands to manage branches of the open repository or
of the repository identified by the -R or --repository option.

   fossil branch new BRANCH-NAME BASIS ?OPTIONS?

       Create a new branch BRANCH-NAME off of check-in BASIS.
       Supported options for this subcommand include:
       --private             branch is private (i.e., remains local)
       --bgcolor COLOR       use COLOR instead of automatic background
       --nosign              do not sign contents on this branch
       --date-override DATE  DATE to use instead of 'now'
       --user-override USER  USER to use instead of the current default

       DATE may be "now" or "YYYY-MM-DDTHH:MM:SS.SSS". If in
       year-month-day form, it may be truncated, the "T" may be
       replaced by a space, and it may also name a timezone offset
       from UTC as "-HH:MM" (westward) or "+HH:MM" (eastward).
       Either no timezone suffix or "Z" means UTC.

   fossil branch list ?-a|--all|-c|--closed?
   fossil branch ls ?-a|--all|-c|--closed?

       List all branches.  Use -a or --all to list all branches and
       -c or --closed to list all closed branches.  The default is to
       show only open branches.

Options:
   -R|--repository FILE       Run commands on repository FILE
```

Figure 93: branch details

5.3.13 merge

This command does the opposite of branch, it brings two branches together.

```
540 ~> fossil help merge
Usage: fossil merge ?OPTIONS? ?VERSION?

The argument VERSION is a version that should be merged into the
current checkout.  All changes from VERSION back to the nearest
common ancestor are merged.  Except, if either of the --cherrypick or
--backout options are used only the changes associated with the
single check-in VERSION are merged.  The --backout option causes
the changes associated with VERSION to be removed from the current
checkout rather than added.

If the VERSION argument is omitted, then Fossil attempts to find
a recent fork on the current branch to merge.

Only file content is merged.  The result continues to use the
file and directory names from the current checkout even if those
names might have been changed in the branch being merged in.

Other options:

  --baseline BASELINE     Use BASELINE as the "pivot" of the merge instead
                          of the nearest common ancestor.  This allows
                          a sequence of changes in a branch to be merged
                          without having to merge the entire branch.

  --binary GLOBPATTERN    Treat files that match GLOBPATTERN as binary
                          and do not try to merge parallel changes.  This
                          option overrides the "binary-glob" setting.

  --case-sensitive BOOL   Override the case-sensitive setting.  If false,
                          files whose names differ only in case are taken
                          to be the same file.

  -f|--force              Force the merge even if it would be a no-op.

  --force-missing         Force the merge even if there is missing content.

  --integrate             Merged branch will be closed when committing.

  -n|--dry-run            If given, display instead of run actions

  -v|--verbose            Show additional details of the merge
```

Figure 94: merge details

5.3.14 tag

This command can be used to control “tags” which are attributes added to any entry in the time line. You can also add/delete/control these tags from the UI by going into the timeline, picking an entry then doing an edit. See Figure **??** on page ??.

```
541 ~> fossil help tag
Usage: fossil tag SUBCOMMAND ...

Run various subcommands to control tags and properties.

    fossil tag add ?OPTIONS? TAGNAME CHECK-IN ?VALUE?

        Add a new tag or property to CHECK-IN. The tag will
        be usable instead of a CHECK-IN in commands such as
        update and merge.  If the --propagate flag is present,
        the tag value propagates to all descendants of CHECK-IN

        Options:
          --raw                       Raw tag name.
          --propagate                 Propagating tag.
          --date-override DATETIME    Set date and time added.
          --user-override USER        Name USER when adding the tag.
          --dryrun|-n                 Display the tag text, but to not
                                      actually insert it into the database.

        The --date-override and --user-override options support
        importing history from other SCM systems. DATETIME has
        the form 'YYYY-MMM-DD HH:MM:SS'.

    fossil tag cancel ?--raw? TAGNAME CHECK-IN

        Remove the tag TAGNAME from CHECK-IN, and also remove
        the propagation of the tag to any descendants.  Use the
        the --dryrun or -n options to see what would have happened.

    fossil tag find ?OPTIONS? TAGNAME

        List all objects that use TAGNAME.  TYPE can be "ci" for
        check-ins or "e" for events. The limit option limits the number
        of results to the given value.

        Options:
          --raw           Raw tag name.
          -t|--type TYPE  One of "ci", or "e".
          -n|--limit N    Limit to N results.

    fossil tag list|ls ?--raw? ?CHECK-IN?

        List all tags, or if CHECK-IN is supplied, list
        all tags and their values for CHECK-IN.

The option --raw allows the manipulation of all types of tags
used for various internal purposes in fossil. It also shows
"cancel" tags for the "find" and "list" subcommands. You should
not use this option to make changes unless you are sure what
you are doing.

If you need to use a tagname that might be confused with
a hexadecimal baseline or artifact ID, you can explicitly
disambiguate it by prefixing it with "tag:". For instance:

  fossil update decaf

will be taken as an artifact or baseline ID and fossil will
probably complain that no such revision was found. However

  fossil update tag:decaf
```

5.3.15 settings

This command is used to set or unset a number of properties for fossil.

```
542 ~> fossil help settings
Usage: fossil settings ?PROPERTY? ?VALUE? ?OPTIONS?
   or: fossil unset PROPERTY ?OPTIONS?

The "settings" command with no arguments lists all properties and their
values.  With just a property name it shows the value of that property.
With a value argument it changes the property for the current repository.

Settings marked as versionable are overridden by the contents of the
file named .fossil-settings/PROPERTY in the check-out root, if that
file exists.

The "unset" command clears a property setting.


   access-log       If enabled, record successful and failed login attempts
                    in the "accesslog" table.  Default: off

   admin-log        If enabled, record configuration changes in the
                    "admin_log" table.  Default: off

   allow-symlinks   If enabled, don't follow symlinks, and instead treat
    (versionable)   them as symlinks on Unix. Has no effect on Windows
                    (existing links in repository created on Unix become
                    plain-text files with link destination path inside).
                    Default: off

   auto-captcha     If enabled, the Login page provides a button to
                    fill in the captcha password.  Default: on

   auto-hyperlink   Use javascript to enable hyperlinks on web pages
                    for all users (regardless of the "h" privilege) if the
                    User-Agent string in the HTTP header look like it came
                    from real person, not a spider or bot.  Default: on

   auto-shun        If enabled, automatically pull the shunning list
                    from a server to which the client autosyncs.
                    Default: on

   autosync         If enabled, automatically pull prior to commit
                    or update and automatically push after commit or
                    tag or branch creation.  If the value is "pullonly"
                    then only pull operations occur automatically.
                    Default: on

   autosync-tries   If autosync is enabled setting this to a value greater
                    than zero will cause autosync to try no more than this
                    number of attempts if there is a sync failure.
                    Default: 1

   binary-glob      The VALUE is a comma or newline-separated list of
    (versionable)   GLOB patterns that should be treated as binary files
                    for committing and merging purposes.  Example: *.jpg

   case-sensitive   If TRUE, the files whose names differ only in case
                    are considered distinct.  If FALSE files whose names
                    differ only in case are the same file.  Defaults to
                    TRUE for Unix and FALSE for Cygwin, Mac and Windows.

   clean-glob       The VALUE is a comma or newline-separated list of GLOB
    (versionable)   patterns specifying files that the "clean" command will
                    delete without prompting or allowing undo.
```

5.4 Miscellaneous

These are commands that don't seem to fit in any category but are useful.

5.4.1 zip

You can do what this command does from the web based user interface. In Figure 13 on page 17 you can download a ZIP archive of the particular version of the files. This command lets you do it from the command line.

```
543 ~> fossil help zip
Usage: fossil zip VERSION OUTPUTFILE [OPTIONS]

Generate a ZIP archive for a check-in.  If the --name option is
used, its argument becomes the name of the top-level directory in the
resulting ZIP archive.  If --name is omitted, the top-level directory
name is derived from the project name, the check-in date and time, and
the artifact ID of the check-in.

The GLOBLIST argument to --exclude and --include can be a comma-separated
list of glob patterns, where each glob pattern may optionally be enclosed
in "..." or '...' so that it may contain commas.  If a file matches both
--include and --exclude then it is excluded.

Options:
  -X|--exclude GLOBLIST   Comma-separated list of GLOBs of files to exclude
  --include GLOBLIST      Comma-separated list of GLOBs of files to include
  --name DIRECTORYNAME    The name of the top-level directory in the archive
  -R REPOSITORY           Specify a Fossil repository
```

Figure 97: zip detail

5.4.2 user

This command lets you modify user information. Again this is a command line duplication of what you can do from the user interface in the browser, see Figure 30 on page 31.

```
544 ~> fossil help user
Usage: fossil user SUBCOMMAND ...  ?-R|--repository FILE?

Run various subcommands on users of the open repository or of
the repository identified by the -R or --repository option.

   fossil user capabilities USERNAME ?STRING?

       Query or set the capabilities for user USERNAME

   fossil user default ?USERNAME?

       Query or set the default user.  The default user is the
       user for command-line interaction.

   fossil user list
   fossil user ls

       List all users known to the repository

   fossil user new ?USERNAME? ?CONTACT-INFO? ?PASSWORD?

       Create a new user in the repository.  Users can never be
       deleted.  They can be denied all access but they must continue
       to exist in the database.

   fossil user password USERNAME ?PASSWORD?

       Change the web access password for a user.
```

Figure 98: user detail

5.4.3 finfo

This command will print the history of any particular file. This can be useful if you need this history in some other system. You can pass this text file to the other system which can than parse and use the data.

```
545 ~> fossil help finfo
Usage: fossil finfo ?OPTIONS? FILENAME

Print the complete change history for a single file going backwards
in time.  The default mode is -l.

For the -l|--log mode: If "-b|--brief" is specified one line per revision
is printed, otherwise the full comment is printed.  The "-n|--limit N"
and "--offset P" options limits the output to the first N changes
after skipping P changes.

In the -s mode prints the status as <status> <revision>.  This is
a quick status and does not check for up-to-date-ness of the file.

In the -p mode, there's an optional flag "-r|--revision REVISION".
The specified version (or the latest checked out version) is printed
to stdout.  The -p mode is another form of the "cat" command.

Options:
  -b|--brief           display a brief (one line / revision) summary
  --case-sensitive B   Enable or disable case-sensitive filenames.  B is a
                       boolean: "yes", "no", "true", "false", etc.
  -l|--log             select log mode (the default)
  -n|--limit N         Display the first N changes (default unlimited).
                       N<=0 means no limit.
  --offset P           skip P changes
  -p|--print           select print mode
  -r|--revision R      print the given revision (or ckout, if none is given)
                       to stdout (only in print mode)
  -s|--status          select status mode (print a status indicator for FILE)
  -W|--width <num>     Width of lines (default is to auto-detect). Must be
                       >22 or 0 (= no limit, resulting in a single line per
                       entry).

See also: artifact, cat, descendants, info, leaves
```

Figure 99: finfo detail

An example would be to run it on the outline.txt file in our book directory:

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil finfo outline.txt
History of outline.txt
2010-05-17 [0272dc0169] Finished maintenance commands (user: jim, artifact:
           [25b6e38e97])
2010-05-12 [5e5c0f7d55] End of day commit (user: jim, artifact: [d1a1d31fbd])
2010-05-10 [e924ca3525] End of day update (user: jim, artifact: [7cd19079a1])
2010-05-09 [0abb95b046] Intermediate commit, not done with basic commands
           (user: jim, artifact: [6f7bcd48b9])
2010-05-07 [6921e453cd] Update outline & book corrections (user: jim,
           artifact: [4eff85c793])
2010-05-03 [158492516c] Moved to clone repository (user: jim, artifact:
           [23b729cb66])
2010-05-03 [1a403c87fc] Update before moving to server (user: jim, artifact:
```

```
          [706a9d394d])
2010-04-30 [fa5b9247bd] Working on chapter 1 (user: jim, artifact:
           [7bb188f0c6])
2010-04-29 [51be6423a3] Update outline (user: jim, artifact: [7cd39dfa06])
2010-04-27 [39bc728527] [1665c78d94] Ticket Use (user: jim, artifact:
           [1f82aaf41c])
2010-04-26 [497b93858f] Update to catch changes in outline (user: jim,
           artifact: [b870231e48])
2010-04-25 [8fa0708186] Initial Commit (user: jim, artifact: [34a460a468])
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

5.4.4 timeline

This prints out the timeline of the project in various ways. The command would be useful if you were building a GUI front end for Fossil and wanted to display the timeline. You could issue this command and get the result back and display it in your UI. There are a number of options in the command to control the listing.

```
547 ~> fossil help timeline
Usage: fossil timeline ?WHEN? ?CHECKIN|DATETIME? ?OPTIONS?

Print a summary of activity going backwards in date and time
specified or from the current date and time if no arguments
are given.  The WHEN argument can be any unique abbreviation
of one of these keywords:

    before
    after
    descendants | children
    ancestors | parents

The CHECKIN can be any unique prefix of 4 characters or more. You
can also say "current" for the current version.

DATETIME may be "now" or "YYYY-MM-DDTHH:MM:SS.SSS". If in
year-month-day form, it may be truncated, the "T" may be replaced by
a space, and it may also name a timezone offset from UTC as "-HH:MM"
(westward) or "+HH:MM" (eastward). Either no timezone suffix or "Z"
means UTC.


Options:
  -n|--limit N         Output the first N entries (default 20 lines).
                       N=0 means no limit.
  -p|--path PATH       Output items affecting PATH only.
                       PATH can be a file or a sub directory.
  --offset P           skip P changes
  -t|--type TYPE       Output items from the given types only, such as:
                           ci = file commits only
                           e  = technical notes only
                           t  = tickets only
                           w  = wiki commits only
  -v|--verbose         Output the list of files changed by each commit
                       and the type of each change (edited, deleted,
                       etc.) after the check-in comment.
  -W|--width <num>     Width of lines (default is to auto-detect). Must be
                       >20 or 0 (= no limit, resulting in a single line per
                       entry).
  -R REPO_FILE         Specifies the repository db to use. Default is
                       the current checkout's repository.
```

Figure 100: timeline detail

5.4.5 wiki

This command allows you to have command line control of the wiki. Again this is useful if you were writing a shell to control Fossil or wanted to add a number of computer generated pages to the Wiki.

```
548 ~> fossil help wiki
Usage: fossil wiki (export|create|commit|list) WikiName

Run various subcommands to work with wiki entries or tech notes.

   fossil wiki export PAGENAME ?FILE?
   fossil wiki export ?FILE? -t|--technote DATETIME|TECHNOTE-ID

      Sends the latest version of either a wiki page or of a tech note
      to the given file or standard output.
      If PAGENAME is provided, the wiki page will be output. For
      a tech note either DATETIME or TECHNOTE-ID must be specified. If
      DATETIME is used, the most recently modified tech note with that
      DATETIME will be sent.

   fossil wiki (create|commit) PAGENAME ?FILE? ?OPTIONS?

      Create a new or commit changes to an existing wiki page or
      technote from FILE or from standard input. PAGENAME is the
      name of the wiki entry or the timeline comment of the
      technote.

      Options:
        -M|--mimetype TEXT-FORMAT   The mime type of the update.
                                    Defaults to the type used by
                                    the previous version of the
                                    page, or text/x-fossil-wiki.
                                    Valid values are: text/x-fossil-wiki,
                                    text/markdown and text/plain. fossil,
                                    markdown or plain can be specified as
                                    synonyms of these values.
        -t|--technote DATETIME      Specifies the timestamp of
                                    the technote to be created or
                                    updated. When updating a tech note
                                    the most recently modified tech note
                                    with the specified timestamp will be
                                    updated.
        -t|--technote TECHNOTE-ID   Specifies the technote to be
                                    updated by its technote id.
        --technote-tags TAGS        The set of tags for a technote.
        --technote-bgcolor COLOR    The color used for the technote
                                    on the timeline.

   fossil wiki list ?OPTIONS?
   fossil wiki ls ?OPTIONS?

      Lists all wiki entries, one per line, ordered
      case-insensitively by name.

      Options:
        -t|--technote               Technotes will be listed instead of
                                    pages. The technotes will be in order
                                    of timestamp with the most recent
                                    first.
        -s|--show-technote-ids      The id of the tech note will be listed
                                    along side the timestamp. The tech note
                                    id will be the first word on each line.
                                    This option only applies if the
                                    --technote option is also specified.

DATETIME may be "now" or "YYYY-MM-DDTHH:MM:SS.SSS". If in
```

5.5 Advanced

These are commands that you will rarely have to use. These are functions that are needed to do very complicated things with Fossil. If you have to use these you are probably way beyond the audience for this book.

5.5.1 scrub

This is used to removed sensitive information like passwords from a repository. This allows you to then send the whole repository to someone else for their use.

```
551 ~> fossil help scrub
Usage: fossil scrub ?OPTIONS? ?REPOSITORY?

The command removes sensitive information (such as passwords) from a
repository so that the repository can be sent to an untrusted reader.

By default, only passwords are removed.  However, if the --verily option
is added, then private branches, concealed email addresses, IP
addresses of correspondents, and similar privacy-sensitive fields
are also purged.  If the --private option is used, then only private
branches are removed and all other information is left intact.

This command permanently deletes the scrubbed information. THE EFFECTS
OF THIS COMMAND ARE IRREVERSIBLE. USE WITH CAUTION!

The user is prompted to confirm the scrub unless the --force option
is used.

Options:
  --force     do not prompt for confirmation
  --private   only private branches are removed from the repository
  --verily    scrub real thoroughly (see above)
```

Figure 102: scrub detail

5.5.2 search

This is used to search the timeline entries for a pattern. This can also be done in your browser on the timeline page.

```
552 ~> fossil help search
Usage: fossil search [-all|-a] [-limit|-n #] [-width|-W #] pattern...

Search for timeline entries matching all words provided on the
command line. Whole-word matches scope more highly than partial
matches.

Outputs, by default, some top-N fraction of the results. The -all
option can be used to output all matches, regardless of their search
score.  The -limit option can be used to limit the number of entries
returned.  The -width option can be used to set the output width used
when printing matches.

Options:

    -a|--all          Output all matches, not just best matches.
    -n|--limit N      Limit output to N matches.
    -W|--width WIDTH  Set display width to WIDTH columns, 0 for
                      unlimited. Defaults the terminal's width.
```

Figure 103: search detail

5.5.3 sha1sum

This can compute the sha1 value for a particular file. These sums are the labels that Fossil uses on all objects and should be unique for any file.

```
553 ~> fossil help sha1sum
Usage: fossil sha1sum FILE...

Compute an SHA1 checksum of all files named on the command-line.
If a file is named "-" then take its content from standard input.
```

Figure 104: sha1sum detail

5.5.4 configuration

This command allows you to save or load a custom configuration of Fossil.

```
555 ~> fossil help configuration
Usage: fossil configuration METHOD ... ?OPTIONS?

Where METHOD is one of: export import merge pull push reset.  All methods
accept the -R or --repository option to specify a repository.

   fossil configuration export AREA FILENAME

        Write to FILENAME exported configuration information for AREA.
        AREA can be one of:  all email project shun skin ticket user

   fossil configuration import FILENAME

        Read a configuration from FILENAME, overwriting the current
        configuration.

   fossil configuration merge FILENAME

        Read a configuration from FILENAME and merge its values into
        the current configuration.  Existing values take priority over
        values read from FILENAME.

   fossil configuration pull AREA ?URL?

        Pull and install the configuration from a different server
        identified by URL.  If no URL is specified, then the default
        server is used. Use the --legacy option for the older protocol
        (when talking to servers compiled prior to 2011-04-27.)  Use
        the --overwrite flag to completely replace local settings with
        content received from URL.

   fossil configuration push AREA ?URL?

        Push the local configuration into the remote server identified
        by URL.  Admin privilege is required on the remote server for
        this to work.  When the same record exists both locally and on
        the remote end, the one that was most recently changed wins.
        Use the --legacy flag when talking to older servers.

   fossil configuration reset AREA

        Restore the configuration to the default.  AREA as above.

   fossil configuration sync AREA ?URL?

        Synchronize configuration changes in the local repository with
        the remote repository at URL.

Options:
   -R|--repository FILE       Extract info from repository FILE

See also: settings, unset
```

Figure 105: configuration detail

5.5.5 descendants

This is used to find where the checked out files are in the time line.

```
556 ~> fossil help descendants
Usage: fossil descendants ?CHECKIN? ?OPTIONS?

Find all leaf descendants of the check-in specified or if the argument
is omitted, of the check-in currently checked out.

Options:
   -R|--repository FILE       Extract info from repository FILE
   -W|--width <num>           Width of lines (default is to auto-detect).
                              Must be >20 or 0 (= no limit, resulting in a
                              single line per entry).

See also: finfo, info, leaves
```

Figure 106: descendants detail

6 Chiselapp

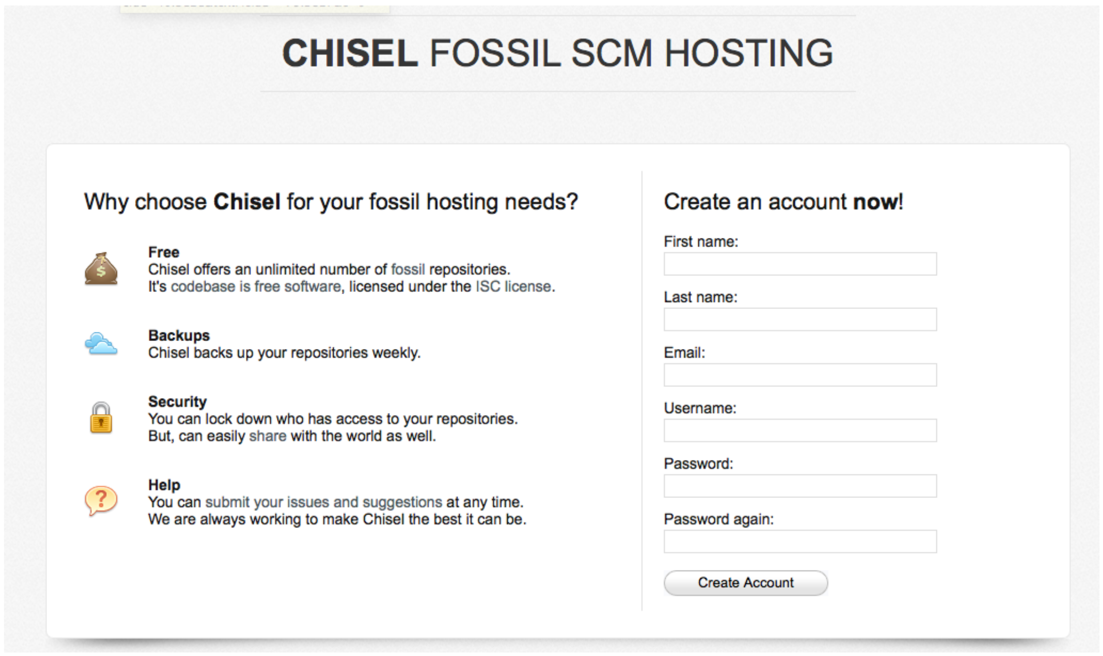
Chiselapp (http://chiselapp.com)) is a website that is like github but hosts Fossil repositories. This way you can have your repository on a internet accessible host. This works like the Apache hosted repositories described in 3.2.1.2 on page 29 but Chiselapp supplies the server and the host is on the internet not a local area network.

After you set up a FREE account you can then push your repository to them and zap you are on the internet at:

```
https://chiselapp.com/user/<your account>/repository/<Project>
```

6.1 Create an account

Your first step is to create an account. The Chiselapp home page is:



Why choose **Chisel** for your fossil hosting needs?

Free
Chisel offers an unlimited number of fossil repositories.
It's codebase is free software, licensed under the ISC license.

Backups
Chisel backs up your repositories weekly.

Security
You can lock down who has access to your repositories.
But, can easily share with the world as well.

Help
You can submit your issues and suggestions at any time.
We are always working to make Chisel the best it can be.

Create an account **now**!

First name:
Last name:
Email:
Username:
Password:
Password again:

Figure 107: Chiselapp Home page

Fill out the form with your information in my case I used my name and my Gmail account to set it up and my account is **jschimpf.**

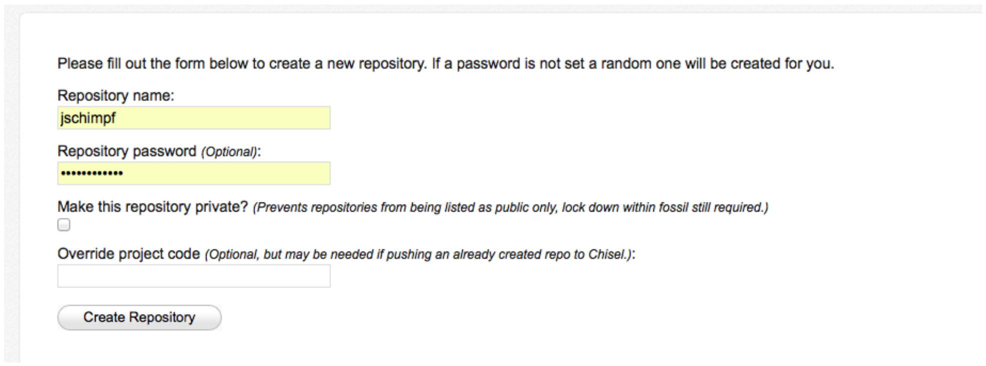
6.2 Repositories

You can create repositories on the site and then copy one of your local repositories there. You have the choice of making public or private repositories. Public are visible to anyone visiting the site and private are visible only to you. In addition you do the standard Fossil assignment of users and privileges so once someone accesses the repository they only can do what you allow.(Figure: 8 on page 13)

The rest of this section will show how I am putting the repository NULLMODEM `http://chiselapp.com/user/jschimpf/repository/NULMODEM/index` on to ChiselApp.

6.2.1 Create Repository

The first step is to pick the option **Create New Repository** on your login page. This will give you the following screen:



Please fill out the form below to create a new repository. If a password is not set a random one will be created for you.

Repository name: jschimpf

Repository password *(Optional)*: ••••••••••••

Make this repository private? *(Prevents repositories from being listed as public only, lock down within fossil still required.)*

Override project code *(Optional, but may be needed if pushing an already created repo to Chisel.)*:

Create Repository

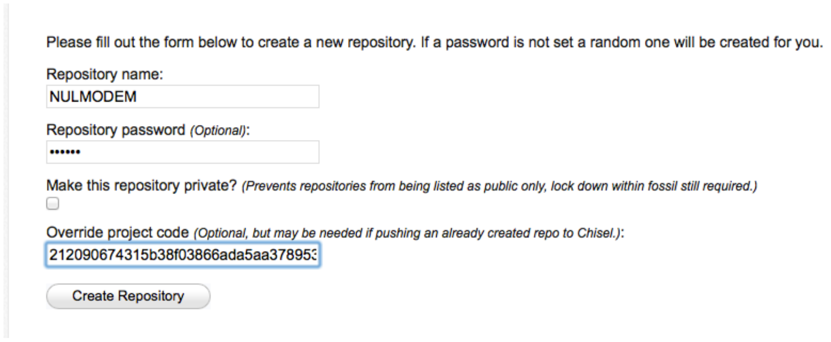
Figure 108: Create New Repository

So I fill in the name as NULLMODEM and I put in my repository password but what is Project Code ? Here you have to run Fossil to extract this information from your repository as follows:

```
500 FOSSIL> fossil info -R NULMODEM.fossil
project-name: NULMODEM
project-code: 212090674315b38f03866ada5aa378953fa9f432
checkins:     6
```

Figure 109: Getting the project code

The form is now filled in and we can create the repository



Please fill out the form below to create a new repository. If a password is not set a random one will be created for you.

Repository name: NULMODEM

Repository password *(Optional)*: ••••••

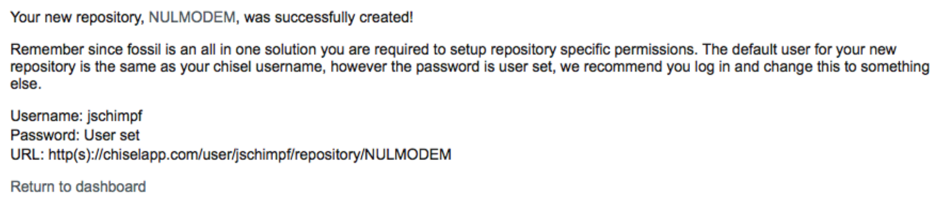
Make this repository private? *(Prevents repositories from being listed as public only, lock down within fossil still required.)*

Override project code *(Optional, but may be needed if pushing an already created repo to Chisel.)*: 212090674315b38f03866ada5aa37895‌3

Create Repository

Figure 110: Filled in form

and you get this:



Your new repository, NULMODEM, was successfully created!

Remember since fossil is an all in one solution you are required to setup repository specific permissions. The default user for your new repository is the same as your chisel username, however the password is user set, we recommend you log in and change this to something else.

Username: jschimpf
Password: User set
URL: http(s)://chiselapp.com/user/jschimpf/repository/NULMODEM

Return to dashboard

Figure 111: Success

6.2.2 Moving data

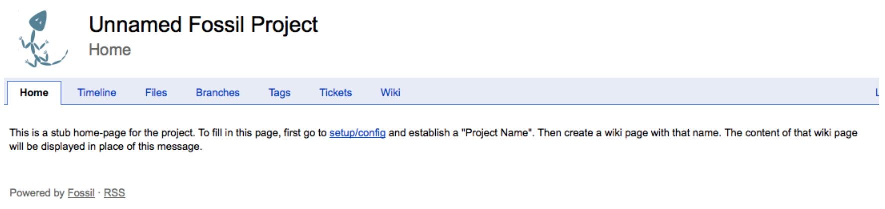
The next step is moving the repository on my disk to Chiselapp. This is done via a push command in Fossil. I am doing this command in the directory where NULMODEM.fossil lives so I don't need to type a path. Note the command is complete but I'm hiding my password when you do this type you password in full where I have <passwd>.

```
501 FOSSIL> fossil push https://jschimpf:<passwd>@chiselapp.com/user/jschimpf/repository/NULMODEM -R NULMODEM.fossil --once
Round-trips: 12   Artifacts sent: 136  received: 0
Push finished with 1149855 bytes sent, 16690 bytes received
502 FOSSIL>
```

Figure 112: Sending the repository

6.2.3 Fixing Data

When you go to your new repository things are a bit messed up. You get:



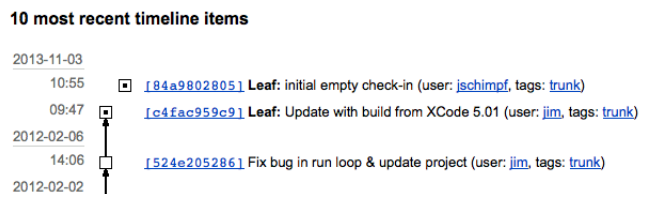
This is a stub home-page for the project. To fill in this page, first go to setup/config and establish a "Project Name". Then create a wiki page with that name. The content of that wiki page will be displayed in place of this message.

Powered by Fossil · RSS

Figure 113: Initial Repository view

Whoa where's all my nice formatting and pointers to my documentation ? They are hidden and you have to get them back:

Go to the timeline view:

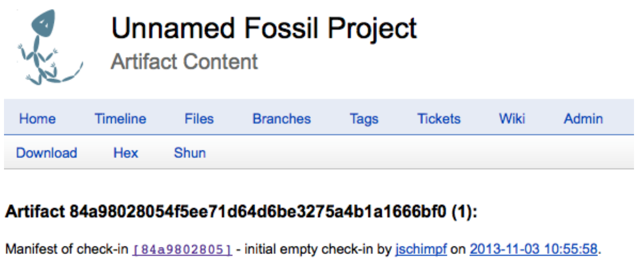


10 most recent timeline items

2013-11-03
10:55 [84a9802805] **Leaf:** initial empty check-in (user: jschimpf, tags: trunk)
09:47 [c4fac959c9] **Leaf:** Update with build from XCode 5.01 (user: jim, tags: trunk)
2012-02-06
14:06 [524e205286] Fix bug in run loop & update project (user: jim, tags: trunk)
2012-02-02

Figure 114: Time Line view

And see the top checkin that is the initial empty check-in this is an artifact of how Chiselapp creates your repository and you have to SHUN it



Unnamed Fossil Project
Artifact Content

Home Timeline Files Branches Tags Tickets Wiki Admin
Download Hex Shun

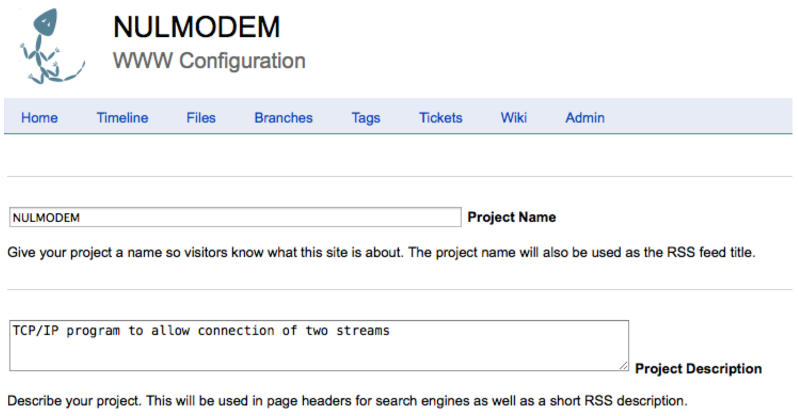
Artifact 84a98028054f5ee71d64d6be3275a4b1a1666bf0 (1):

Manifest of check-in [84a9802805] - initial empty check-in by jschimpf on 2013-11-03 10:55:58.

Figure 115: Shun initial check-in

You will then be taken to another page where it will ask you if you really want to do this and pick Shun again.

Not quite there yet, you have to log into the project (Remember your name and password from Figure: 110 on page 101) log in with this information and go to the Admin->Configuration page. Put in the same information you had on your local repository and ZAP your home page is back.



NULMODEM
WWW Configuration

Home Timeline Files Branches Tags Tickets Wiki Admin

NULMODEM **Project Name**

Give your project a name so visitors know what this site is about. The project name will also be used as the RSS feed title.

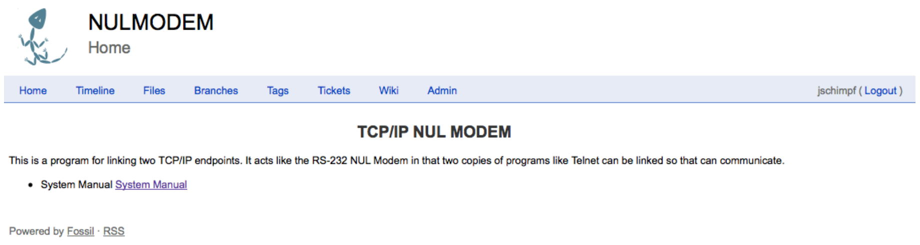
TCP/IP program to allow connection of two streams **Project Description**

Describe your project. This will be used in page headers for search engines as well as a short RSS description.

Figure 116: Set Configuration

6.2.4 Final Fixes

The home page is now restored and we are ready to go.



TCP/IP NUL MODEM

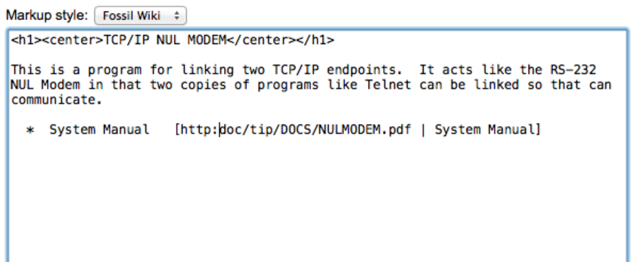
This is a program for linking two TCP/IP endpoints. It acts like the RS-232 NUL Modem in that two copies of programs like Telnet can be linked so that can communicate.

- System Manual System Manual

Powered by Fossil · RSS

Figure 117: Home page fixed

The only problem now is the System Manual link doesn't work. The original was:



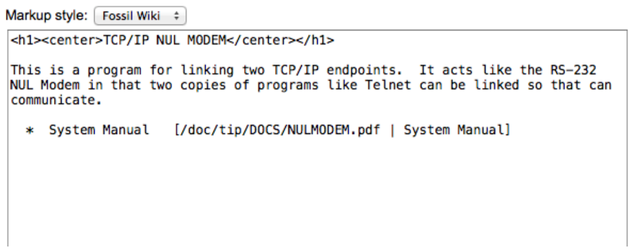
```
<h1><center>TCP/IP NUL MODEM</center></h1>

This is a program for linking two TCP/IP endpoints.  It acts like the RS-232
NUL Modem in that two copies of programs like Telnet can be linked so that can
communicate.

  *  System Manual   [http:doc/tip/DOCS/NULMODEM.pdf | System Manual]
```

Figure 118: Orignal Link

The fix is to change the link to just /doc/tip/DOCS/NULMODEM.pdf



```
<h1><center>TCP/IP NUL MODEM</center></h1>

This is a program for linking two TCP/IP endpoints.  It acts like the RS-232
NUL Modem in that two copies of programs like Telnet can be linked so that can
communicate.

  *  System Manual   [/doc/tip/DOCS/NULMODEM.pdf | System Manual]
```

Figure 119: Fixed link

6.2.5 Synching

When you created the repository on ChiselApp you used this command: (note these lines are just folded to fit the page when used they should all be one line.)

```
fossil push https://jschimpf:<passwd>@chiselapp.com/user/jschimpf/repository/NULMODEM
                                  -R NULMODEM.fossil --once
```

when you did that the repository was created but it did not sync with your local one. This is probably not a good idea as you want the ChiselApp repository to stay up to date. If you leave off the –once then it will sync locally. If it isn't synching now and you want to reverse this at any time just type:

```
fossil push https://jschimpf:<passwd>@chiselapp.com/user/jschimpf/repository/NULMODEM
                                  -R NULMODEM.fossil
```

i.e. the command without the –once and you are syncing again.

6.3 Final Result

Now you can go to https://chiselapp.com/user/jschimpf/repository/NULMODEM and view the repository and do what an anonymous user can do.

7 What's next ?

This book so far has covered how to use the many features of Fossil and has, I hope, interested you in using it. The question "what's next" now comes up. First go to the Fossil website http://www.fossil-scm.org/. While there you can go to the Wiki link and then list all Wiki pages. There are all sorts of topics covered there in depth. If that still doesn't help, you can join the Fossil mailing list (see Wiki links) and look at the archives or directly ask a question. I have found the list to be very helpful and have had my questions asked very quickly.

On the mailing lists you will see long discussions of changes to be made to Fossil, some of these are accepted very quickly and will appear within hours in the Fossil source code. Others engender long discussions (in particular discussion of changes to the Wiki) and it is interesting to read the pros and cons of suggested changes.

Fossil is an evolving program but if you get a version that has all the features you need you can stick with that version as long as you like. Going to a new version though is simple and just requires a **rebuild** of your current repositories. The developers have been very careful to preserve the basic structure so it is easy and safe to switch versions.

Finally if you wish to contribute to the project there are many things to do (See the To Do List in the Wiki).

Index

References

[1] D. Crockford. The application/json media type for javascript object notation. Request for Comments 4627, Network Working Group, July 2006.

[2] Mattias Ettrich. Lyx - the document processor.

[3] Dick Grune. Concurrent versions system, a method for independent cooperation. *unpublished*, 1986.

[4] D. Richard Hipp. Fossil home page.

[5] University of Texas. How we use sccs.

[6] Marc J Rochkind. The source control system. *IEEE Transactions on Software Engineering*, SE-1(4):364, December 1975.