Fossil版本控制
用户向导

Jim Schimpf 著

lomatus 译

文章编号: PAN-20100424

联系作者:

Pandora Products
215 Uschak Road
Derry, PA 15627
United States of America

电话: 724.539.1276
传真: 724.539.1276
网站: http://www.fossil-scm.org/schimpf-book/index
邮箱: jim.schimpf@gmail.com

联系译者:

邮箱: lomatus@163.com

当前版本:

版本号: 2.0

日期: 2012-11-29

翻译版本:

译文版本: 1.0

日期: 2013-09-18

Contents

1. 源码控制 & 为什么需要她 **1**

1.1. 什么是源码控制? 1

1.2. 为何在项目里使用文档版本? 2

1.2.1. 如何获取 2

1.3. 源码控制描述 2

1.3.1. Check Out 检入检出系统 2

1.3.2. Merge 集中式系统 3

1.3.3. Distributed 分布式系统 3

1.3.4. 一般名词 3

2. 单一用户 **5**

2.1. 简介 5

2.2. 创建一个仓库 5

2.2.1. 简介 5

2.2.2. 创建仓库 5

2.2.3. 连接一个仓库 6

2.2.4. 增加和初始提交 6

2.2.5. Fossil开始使用概要 7

2.3. 设置用户界面 7

2.3.1. 用户界面概要 10

2.4. 更新仓库 11

2.4.1. 更新概要 13

2.5. Ticket-标签 14

2.5.1. Ticket-标签概要 18

2.6. Wiki的使用 19

2.6.1. Wiki概要 22

3. 多用户模式 **23**

3.1. 简介 23

3.2. 设置 23

3.2.1. 服务器设置 23

3.2.1.0.1. 自托管 23

3.2.1.0.2. 服务器托管 24

3.2.1.0.3. 托管服务器的CGI脚本 24

3.2.2. 测试(含自托管和服务器托管) 25

3.3. 用户账户 25

3.4. 多用户运行 27
3.4.1. 克隆 27
3.4.2. 保持同步 28
3.4.3. 复杂化 29
3.4.4. 修复更新文件 31
3.4.5. 修复合并文件 33

4. 分叉 & 分支 **35**
4.1. 简介 35
4.2. 分叉，分支& 合并 35
4.2.1. Marilyn的操作 35
4.2.2. Jim的操作 36
4.2.3. 修复分叉fork 37
4.2.4. 使用的命令 39
4.3. 不合并的分叉 40
4.3.1. 检入（Checkin）企图 40
4.3.2. 更新 40
4.3.3. 使用的命令 41
4.4. 分支Branch 42
4.4.1. 简介 42
4.4.2. 将仓库添加分支 42
4.4.3. 颜色设置 43
4.4.4. 检出分支 44
4.4.5. 修复错误(全部) 46
4.4.6. 使用的命令 47

5. Fossil命令 **49**
5.1. 简介 49
5.2. 基本命令 49
5.2.1. help 49
5.2.2. add 50
5.2.3. rm or del 50
5.2.4. rename or mv 51
5.2.5. status 51
5.2.6. changes 52
5.2.7. extra 52
5.2.8. revert 53
5.2.9. update 53
5.2.10. checkout or co 54
5.2.11. undo 54
5.2.12. diff 54
5.2.13. gdiff 55
5.2.14. ui 55
5.2.15. server 56
5.2.16. commit or ci 56

5.3. 维护命令 . . . 57
5.3.1. new . . . 57
5.3.2. clone . . . 58
5.3.3. open . . . 59
5.3.4. close . . . 59
5.3.5. version . . . 59
5.3.6. rebuild . . . 60
5.3.7. all . . . 60
5.3.8. push . . . 60
5.3.9. pull . . . 61
5.3.10. sync . . . 61
5.3.11. clean . . . 62
5.3.12. branch . . . 62
5.3.13. merge . . . 63
5.3.14. tag . . . 63
5.3.15. settings . . . 64
5.4. 杂项 . . . 65
5.4.1. zip . . . 66
5.4.2. user . . . 66
5.4.3. finfo . . . 66
5.4.4. timeline . . . 67
5.4.5. wiki . . . 68
5.5. 高级命令 . . . 69
5.5.1. scrub . . . 69
5.5.2. search . . . 69
5.5.3. sha1sum . . . 69
5.5.4. rstats . . . 70
5.5.5. configuration . . . 70
5.5.6. descendants . . . 71

6. Fossil定制 - TH脚本语言 73
6.1. TH简介 . . . 73
6.1.1. TH是一个类Tcl语言 . . . 73
6.2. “Hello, world”程序 . . . 73
6.3. TH结构和语法 . . . 73
6.3.1. 数据类型 . . . 73
6.3.2. 列表 . . . 74
6.3.3. 命令 . . . 74
6.3.4. 分割 & 转义 . . . 75
6.3.4.1. 参数分割 . . . 75
6.3.4.2. 值转义 . . . 75
6.3.4.3. 反斜杠转义 . . . 76
6.3.4.4. 变量转义 . . . 76
6.3.4.5. 命令转义 . . . 77

6.3.4.6. 再述参数分隔 77
6.3.5. 概要 78
6.3.5.1. 警告 78
6.4. TH表达式 79
6.5. TH变量 80
6.5.1. 使用变量 81
6.5.1.1. 标量变量和数组变量 81
6.5.1.2. 变量范围 82
6.6. TH命令，脚本和程序流程 82
6.6.1. 再述命令 82
6.6.2. 脚本 83
6.6.3. 命令结果代码 83
6.6.4. 流程控制命令 84
6.6.5. 创建用户自定义命令 85
6.6.6. 执行用户自定义命令 86
6.6.7. 特殊命令 86
6.7. 使用字符串 87
6.8. 使用列表 88

7. 下面做什么? **89**

A. 修订记录 **97**

免责声明

潘多拉产品公司已经仔细地检查该文档的信息并且相信它是准确的。但是，我们申明对该文章的可能包含的不准确之处不包含任何责任。在任何情况下潘多拉对直接、非直接、特殊的、典型的、偶然的或者间接的由于缺陷、疏漏造成的损害结果不承担相关责任，甚至包含是由于建议所可能造成的损害。

在产品的研发中，潘多拉保留在不通知及履行相关义务的情况下提高该文档信息准确性以及更改产品描述的权利。

其它贡献者名单

- Marilyn Noz - 编辑
- Wolfgang Stumvoll - Fossil合并及Windows的使用
- Paul Ruizendaal - TH 1 脚本语言手册
- javalinBCD@gmail.com - 发现BUG
- arnel_legaspi@msn.com - 发现BUG
- lomatus@163.com - 中文版翻译

1. 源码控制 & 为什么需要她

1.1. 什么是源码控制?

一个源码控制系统是一个用于管理项目内文件的软件。项目(例如该书或者一个软件应用)一般包含了大量的文件。这些文件依次被组织在文件夹或者子文件夹内进行管理。在任何一个特定的时间内这些文件在编辑状态下由项目的最新版本的（文件）工作拷贝组成。如果其它的人员也在该项目里进行工作你必须将你的当前正在工作复制给他们以让他们进行工作。当他们查找并修改问题后，他们的工作拷贝将与你的工作拷贝不一样（它在一个不一样的版本里）。当你可以继续他们的工作时，你需要某种机制更新你的这些文件的工作拷贝以同步到团队工作的最新版本。一旦你更新了你的文件，这个新版本的项目还将再次执行同样的循环周期。就像是当前的版本会被编一个唯一的号码，这就是为什么书有版本软件也有版本号的原因。

当软件开发工程师在一个大型项目里和多个开发人员一起时会有这个周期且意识到他们需要一个工具来控制这些变更。当多个开发人员开发时，同一个文件将会被二个人修改并且是不一样的方向，什么被修改的记录也会丢失。在软件发布一个新版本时将会很难保证所有团队成员的BUG修复和功能增强都被包含在内。

一个叫The Source Control System[6]的源码控制系统在1972年由Bell Labs开发出来用于跟踪文件修改。它会记录对每一个文件的修改到一个文件内且包含关于这个修改的描述（为什么这样修改）。它同样也限制了谁可以对文件进行修改以保障不会发生编辑文件的冲突发生. [5]

这个很重要，但是开发者们会发现他们需要更多的功能。他们需要可以保存所有项目里文档的状态并且给他们一个名字（例如：Release 3.14）。当软件项目成熟时你将不但会有一个发布版本来使用还有一个BUG报告来针对该版本，当下一个版本发布时会增加新的功能并修复之前的BUG。源码控制系统将不得不去处理这些现在叫做分支的问题。一个分支例如叫做“版本2”发布了但是却在继续修复比如版本1.1,1.2的问题。同时，你还有另外一个分支叫做“版本2”正在进行添加新功能的编码工作。

在1986年开放源代码的并发版本控制系统CVS [3] 被开发出来。这个系统允许对一组文件添加标签而且允许同时拥有多分支（例如：版本）。当时有许多其它的版本控制系统被开发出来，有开源的也有封闭的。

在2006年发布 [4] 的Fossil是一个安装简单的版本控制系统同时还集成了问题标签系统(图解 2.17 位于 16页) ，一个wiki (图解 2.6 位于 19页) 还有内建一个网站服务器 (图解 2.5 位于 8页)。

1.2. 为何在项目里使用文档版本?

你为什么需要使用一个源码控制系统? 你无法在文件夹间随意地创建文件，删除文件，移动文件。在你的代码里进行修改时有一个需要仔细遵照的检查清单和步骤。

有这么多激辩，为什么还要做? 作为一个开发人员最可怕的噩梦是 “它昨天还工作得很好” 综合症。是的，你已经使它工作起来了但是现在又不行了。如果你只是在一个文档内工作，可以想象的是你保存了不同名词(可能是日期后缀)的副本，因此你可以回退回去。但是如果你没有这样做，毫无疑问，你无法恢复到你之前的代码。如果有源码控制系统且坚持遵循其规程，你可以立即回退到昨天某个时间的代码，或者是一个小时之前的，甚至是上个月的。在你完成这些后，你开始知道什么是好的代码，你可以检查出到底发生了什么（才导致这样的结果）。

在有源码控制系统的的帮助下，你还可以自由地进行试验，“让我们试试那个激进的新技术”，如果它无法工作，那么就退回到之前的状态。

这本书的其余部分是一个Fossil版本控制系统的用户手册，用以进行代码管理甚至更多更多。它可以运行在多个操作系统之上且免费、开放源代码(FOSS协议)。它安装非常简单只有一个可执行文件，且代码仓库只有一个文件，可以非常简单地备份并且大约只有源代码50% 的大小（SQLite数据库压缩）。

1.2.1. 如何获取

如果这些可以激起你的兴趣，你可以从http://www.fossil-scm.org/download.html获得一个Fossil可执行行文件的拷贝。该页面里有Linux、Mac和Windows系统的可执行文件的链接。如果你想要自己进行编译也可以找到源代码。该网站还包含这个PDF和这个PDF的源代码，该网站托管使用的是Fossil (查看章节 3 位于 23页).

1.3. 源码控制描述

如果你没有使用过源码控制系统，那么该章节将非常有用。我将定义一些词汇并且解释下源码控制的基本理念。

1.3.1. Check Out 检入检出系统

当描述源码控制的源祖如SCCS时我说过它管理一个单一文件的变更且阻止多个人同时在同一个文件上工作。这是一个典型的全局源码控制系统。这样你可以理解“检入（check out）”一个文件并编辑它，同时同样使用该源码控制系统的人可以看到谁（也就是你）在这个文件上进行工作那么他们被禁止碰这个文件。他们可以获得一个只读的文件拷贝，那么他们说他们可以构建软件但除了那个文件，因为只有“所有者”可以编辑它。当完成编辑后，“所有者”检回那个文件，然后其它任何一个人就可以继续在该文件上工作。检回的同时系统记录了谁曾经编辑了它以及做了什么修改。

这个系统在一个可以实时沟通小的组织里工作得很好，一个常见的问题是一个文件被某个人检出了，而**你** 必须对它进行修改，在一个小团队的组织中，你只要对着房间大喊一下就可以解决这个问题。

1.3.2. Merge 集中式系统

在以CVS或Subversion为代表的系统中需要区分的是不是获取文件然后编辑而是将其推送到版本控制中去。在这些系统中你将源代码文件放到一个你本机的工作文件夹内，然后你编辑这些文件并作必要的修改，当完成时你提交或者将它们检回到代码仓库里去。如此它们回到版本控制系统中且系统知道当前版本和上一个版本之间的变化。

这个解决了上面提到的当其他人锁定的文件无法编辑的问题。你现在解决了这个问题，很多人可以同时在同一个文件上工作并且做出修改。它使用Check-In(检入)流程来处理。在某个时间点只有一个人可以Check-In一个文件。将要发生的是系统检查该文件，如果代码仓库里的文件有修改而不是你之前检入的那个文件，系统将终止检入流程，系统将会询问用户是否想要合并那些新的修改到他的拷贝中去，一旦完成操作，那么最新的版本就可以被成功检入。

这种类型的系统使用在大型项目里如Linux内核开发或者其它拥有大量不同区域分布的贡献者的项目里。

1.3.3. Distributed 分布式系统

目前为止上面我们提到的二个主要系统都是中心化的。也就是说只有一个代码仓库位于一个独立的服务器上，当你获取文件的拷贝或者检入那些文件时，它们都汇集到一个地方。这样的方式可以支持很多很多用户。一个分布式的系统是一个该类系统的扩展，它允许代码仓库分布式存在（有多个仓库）且有使它们同步的机制。

在一个独立服务器上的唯一代码仓库工作模式下，必须可以连接到该服务器以获取更新及检入最新的代码。如果你无法连接到服务器那么你就被困住了而无法继续工作。分布式系统允许你拥有自己的代码仓库拷贝然后可以在离线状态下继续工作,当你可以与服务器进行通讯时你可以与之同步。当人们把工作带回家或者无法连接到公司网络是这个就变得非常有用，每个人都可以拥有一个代码仓库可以离线工作，当回到办公室时可以重新工作内容同步到服务器。

1.3.4. 一般名词

下面是当我谈论版本控制或者Fossil时需要使用的名词的一份清单。

Repository 这个是一个保存版本控制文件的仓库，它由源码控制系统管理。

SCS 源码控制系统，这是一个管理一组文件并跟踪修改，允许多个用户在一个控制的规则规范下编辑它们的软件。

Commit 在Fossil里保存当前的新文件或者变更的文件到代码仓库里去。

Trunk Fossil代码仓库里的代码发展的主要线路（主分支）。

Branch 一个用户定义的代码分支并有SCS提供服务，它允许在一个代码仓库里有多个工作点，老分支（版本）可能有已经结束的BUG修复，新的分支（版本）可能含有添加的新功能。

Fork 在Fossil里是一个在代码路径里的自然而然的分叉（裂隙），当仓库里一个有修改的文件而没有被提交的时候发生。**

2. 单一用户

2.1. 简介

如果你之前阅读过该部分或者被说服来尝试一下，你会想要将你的项目放到Fossil的软件控制之中。这个章节的目的是带领你完成这个任务且想你暂时如何适应在使用该工作来进行开发工作。该章节的假设是你这个仓库的唯一用户，你是这个项目的设计者、开发者和维护者。当你很舒适地使用这个工具后，下一个章节将会想你展示如何在多个用户在同一个项目上进行工作时如果使用它。

2.2. 创建一个仓库

2.2.1. 简介

在“吃狗粮”（俚语：软件公司俚语，指用自己的产品本身来说明产品的质量和能力）精神下，我们将使用这个本作为一个使用Fossil来管理的项目。这本书是由一个文件夹内的文本文件组成(我们使用的工具是： LyX [2])，我的工作目录结构如下:

```
FOSSIL/
    这个文件夹保存了我所有的Fossil仓库
FossilBook/
    outline.txt    - 书的设计
    fossilbook.lyx - 书本身
    Layout
        fossil.png  - Fossil的logo (标题页的图片)
    Research
        fossilbib.bib - 使用的参考文献
        History
            CVC-grune.pdf - CVS的文献
            RCS-A System for Version Control.webloc - RCS 书签
            SCCS-Slideshow.pdf - SCCS的文献
            VCSHistory -pysync ... .webloc - 版本控制历史
```

这本书一个小时前或者刚开始准备Research并建立一个框架，我今天要做的就是建立一个仓库并把所有这些文件都纳入Fossil的控制里。

2.2.2. 创建仓库

我有一个叫做FOSSIL文件夹存放所有的代码仓库，Fossil不在乎但是我这么做的原因是我可以备份他们。首先，我需要为这本书新建一个仓库，可以在我移动到书的文件夹目录后使用下面的命令来实现。

```
[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 2.1.: 创建仓库

我创建了一个后缀为.fossil的仓库，这个当使用服务器时会很有用(见图解 5.16 位于 56页)。在创建过程在他指派了一个管理员账户“Jim”及初始密码。

2.2.3. 连接一个仓库

仓库建好了，但是它是空的且与书的文件夹没有任何联系，下一步就是打开仓库到书的文件夹目录，使用命令**open**。

```
[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 2.2.: 打开仓库 & 检查

open 命令好像什么也没有做，当执行检查命令**status**后，显示了仓库的信息，我们已经将文件夹目录挂载到了存储的主线。

命令**extra**显示了文件夹内所有没有在Fossil控制下的文件，目前我们还没有检入任何东西。

2.2.4. 增加和初始提交

我比如使用**add**命令将所有有用的文件增加到仓库离去。Fossil将自动递归地添加，所以如果我增加顶级目录的文件，那么它递归地进入子文件夹比如Layout和Research目录并将其中的文件也增加到仓库里去。在你执行Add时它会整理你的文件夹目录所以你不会意外地添加了一堆临时文件（例如编译出来的Obj文件）。以后你可以很简单地删除它们但是在动手前的整理工作将会节省掉你的一些工作。

```
[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 2.3.: 初始文件添加

我简单地告诉Fossil添加当前文件夹(.)那么它添加所有的文件以及所有的子文件夹的文件。注意_FOSSIL_文件不会被添加进去。这个是标志文件那么Fossil就会知道这个文件夹属于哪个仓库。Fossil不会加载这个文件直到需要管理它时，但是其余的都会添加。

今天我要做的最后一件事是添加这些文件或者当我提交时添加进去，这一步必要要做。然后让Fossil去管理这些事情，我们就可以放轻松了。

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

Figure 2.4.: 初始提交

我添加了一个备注，一直这样做是一个很好的主意。以后当我们查看这些提交的时间轴时我们就知道我们到底做了些什么事情。

2.2.5. Fossil开始使用概要

- **fossil new <name>** 创建一个新的Fossil仓库
- **fossil open <repository>** 当在一个源代码文件夹时连接这个文件到Fossil仓库
- **fossil add .** 将会添加(递归地)当前文件夹及子目录下所有的文件到仓库里
- **fossil addremove** 将会(递归地)添加任何当前工作目录下的文件或者是删除任何当前工作目录下的文件
- **fossil commit -m "Initial Commit"** 将会提交所有已添加的文件到仓库里去

2.3. 设置用户界面

Fossil的一个令人惊讶的功能是网站服务器。它使你可以在不用任何操作系统特殊配置的情况下拥有一个GUI类型的用户界面，它呈现在你操作系统的网页浏览器内。在上一步中我将项目检入了一个Fossil仓库，下面我将准备一个网页界面来支持生产。

注意: Fossil使用8080替代默认的的80标准端口作为所有HTTP连接的端口。运行后它将自动地启动你的浏览器并打开仓库的主页。不幸的是我的这个书所用的系统已经被Apache占用了8080端口，所以我使用8081端口。我将一直添加一个额外的参数来运行UI命令。

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

Figure 2.5.: 开启网站服务器

注意: 新版本的Fossil会自动找到一个可用的端口并告诉你它所使用的端口号。你任然可以使用 -port 参数来控制这个端口 #。

这里你可以看到他如何启动，你也可以看到我选用了8081端口，因为你我的8080端口被锁定了。当我运行命令后我的浏览器启动了并看到如下主页。

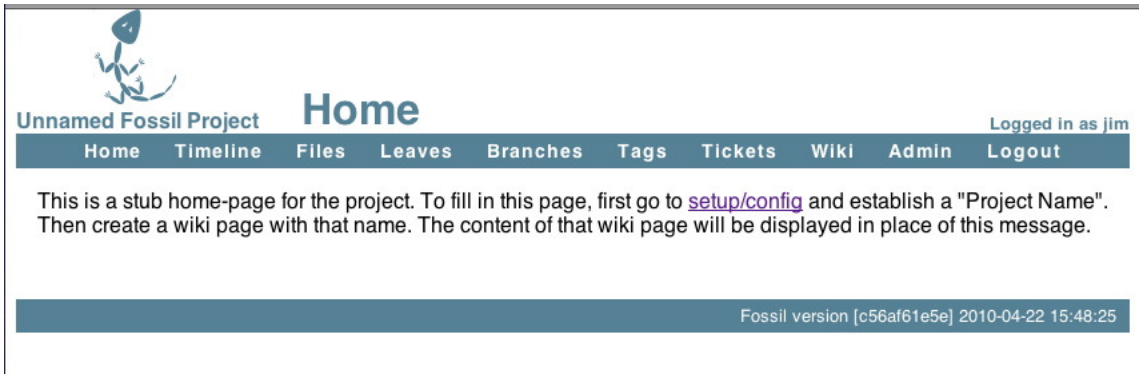

Figure 2.6.: 初始化网页服务器页面

遵循该页面的建议，我跳转到**setup/config**页面。我将会进行最小的设置而你可以对真个项目进行设置。当你对Fossil足够熟悉后，你可能会设置越来越多的选项以适合你的胃口，但是下面的事情是你必须要做的 。

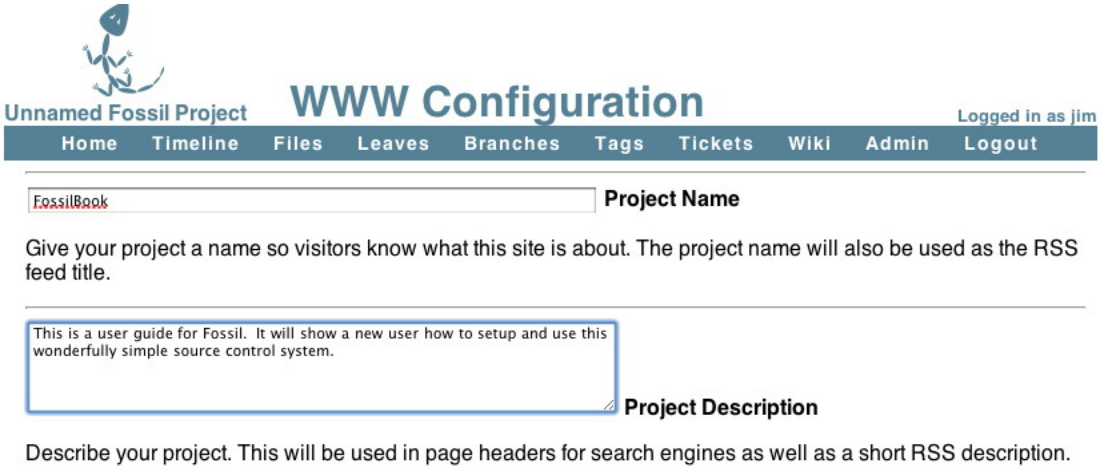

Figure 2.7.: 初始化配置

我输入了一个项目的名称以及相关描述，项目的名称同样会被用来初始化Wiki页面的名称 (see 2.6 位于 19页) 而描述对于其它人来说可以告诉他们你在这里做什么。然后我点击页面的底部 **应用修改**按钮。

下一步我点击了Admin面板(你可以在页面头部找到它)并点击了该页面的用户表单，我呈现在用户里并可以使用这个账户设置项目的密码。

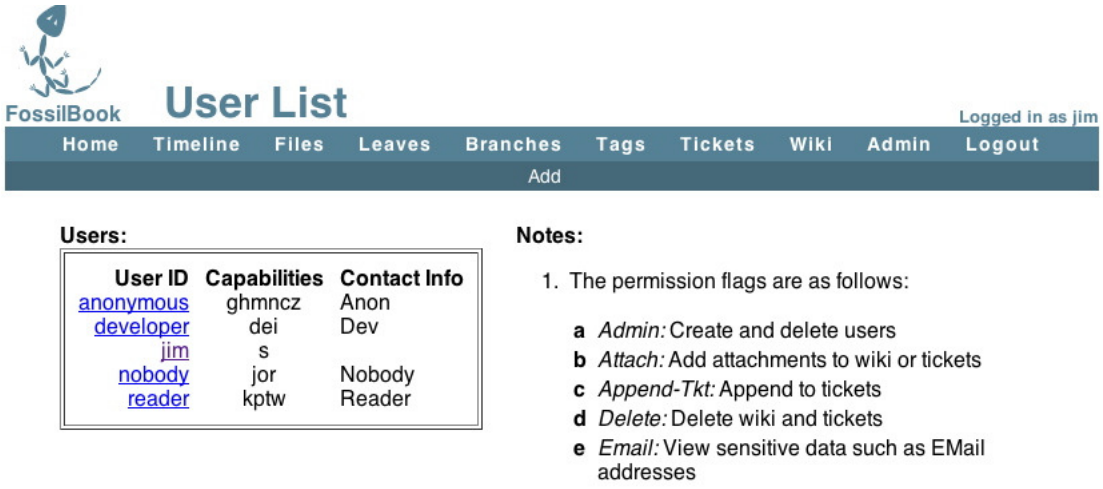

Figure 2.8.: 用户配置

如图Fossil自动配置了一些用户除了项目的创建者。如你所见匿名用户可以在Fossil的网站下载代码。这个用户可以查看获取代码但是无法提交代码。在页面的右侧有很多选项，你可以给一个用户相应权限，当你设置你仓库时认真阅读它们会很有价值。重要是我(Jim)权限是“s”就是超级用户权限。这意味着我可以对这个仓库做任何事情。

我将会编辑用户Jim以保障该账户拥有所有我想要的权限。这里你必须设置密码。还记得你初始化仓库时系统给你自动指派的密码么(图解 2.1 位于 6页), 把它改成一个你可以记得住的密码很有必要而不是使用那个初始密码。

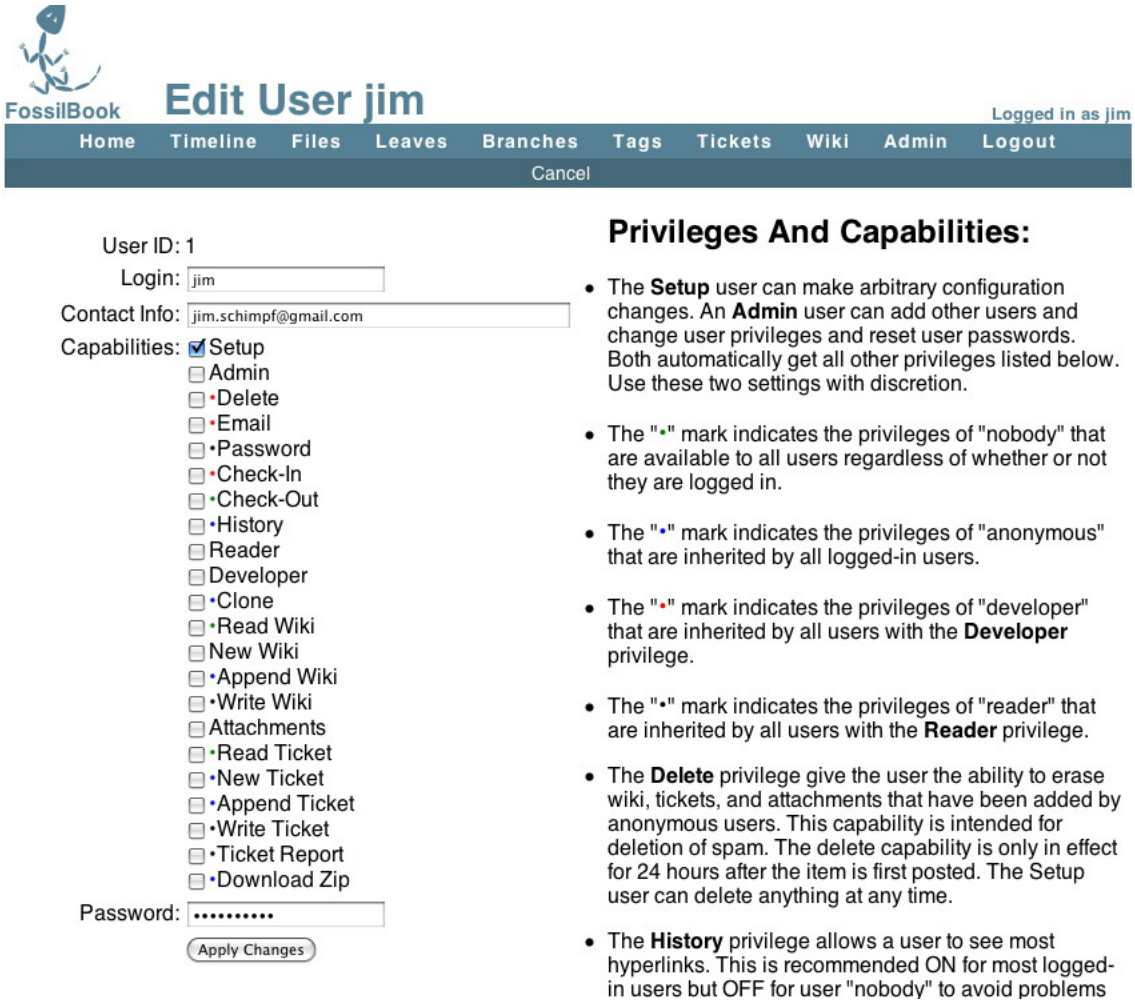

Figure 2.9.: 超级用户设置

我输入了我的联系信息（电子邮箱地址）和一个我可以记得住的密码，当然你无法看到它。然后点击了应用修改。

现在仓库已经准备好进行下一步的工作了，它仅是一个必要的准系统但是最重要的事情已经设置完成了。

2.3.1. 用户界面概要

- **fossil ui** 在源代码目录里运行将会启动基于浏览器的Fossil用户界面

- **fossil ui -port <IP port #>** 可以在8080端口被占用是指定一个端口
- 第一次运行时设置项目名称和密码是很重要的

2.4. 更新仓库

在写本书上面章节的时候我已经创建了一堆新文件且一些已经在仓库里的文件进行了修改。在结束今天的工作前，我应该将新文件添加到仓库里且将更改提交到仓库以保存这个项目的里程碑。

```
[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 2.10.: 项目状态

我运行了**fossil extra**命令查看这些新文件。我想要增加那些图片文件(在文件夹: Images/Single_user)以及另外二个文件#fossilbook.lyx# 和 fossilbook.lyx~，而不想增加LyX的临时文件。我可以运行**fossil status**命令。该命令显示了所有已经在仓库里的文件。发生变化的文件只有文本文件 **fossilbook.lyx**.

我现在要做的就是增加那些图片到仓库再提交变更到仓库，然后我就可以继续我今天要做的其它任务了。

```
[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 2.11.: 更新新文件

在完成这个提交后我可以开启Fossil的UI(见图示 2.5 位于 8页)并点击Fossil UI顶部的Timeline检查项目的时间线。如下图所示:

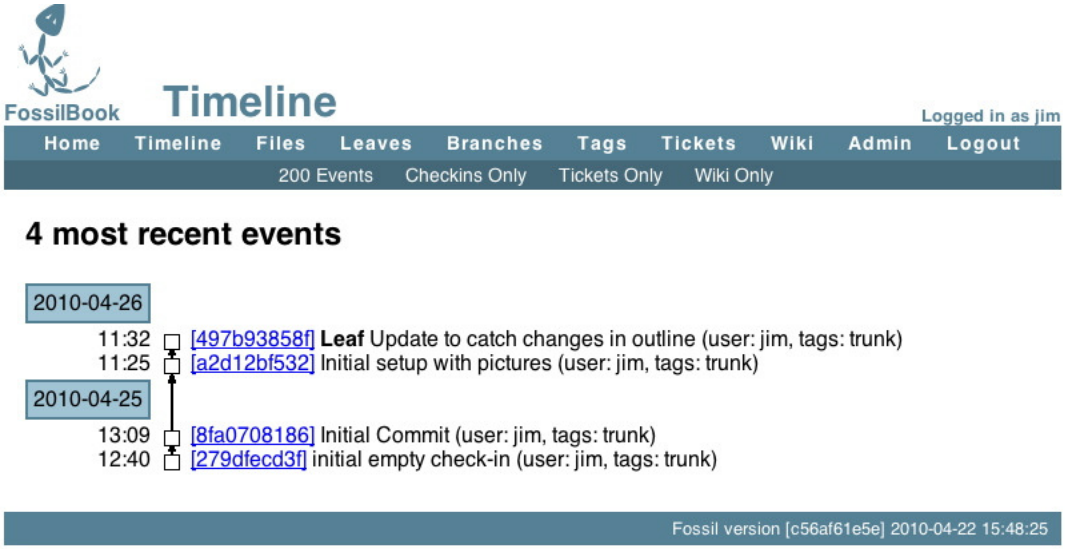

Figure 2.12.: 时间线

你可以看到目前为止所有的检入操作，你还可以看到图示 2.11 on the preceding page后的检入记录。我做了另外一个检入因为我在大纲部分丢失了部分修订。检入记录被十位的哈希数字备注且其是一个你可以点击查看详细版本变更信息的动态链接。

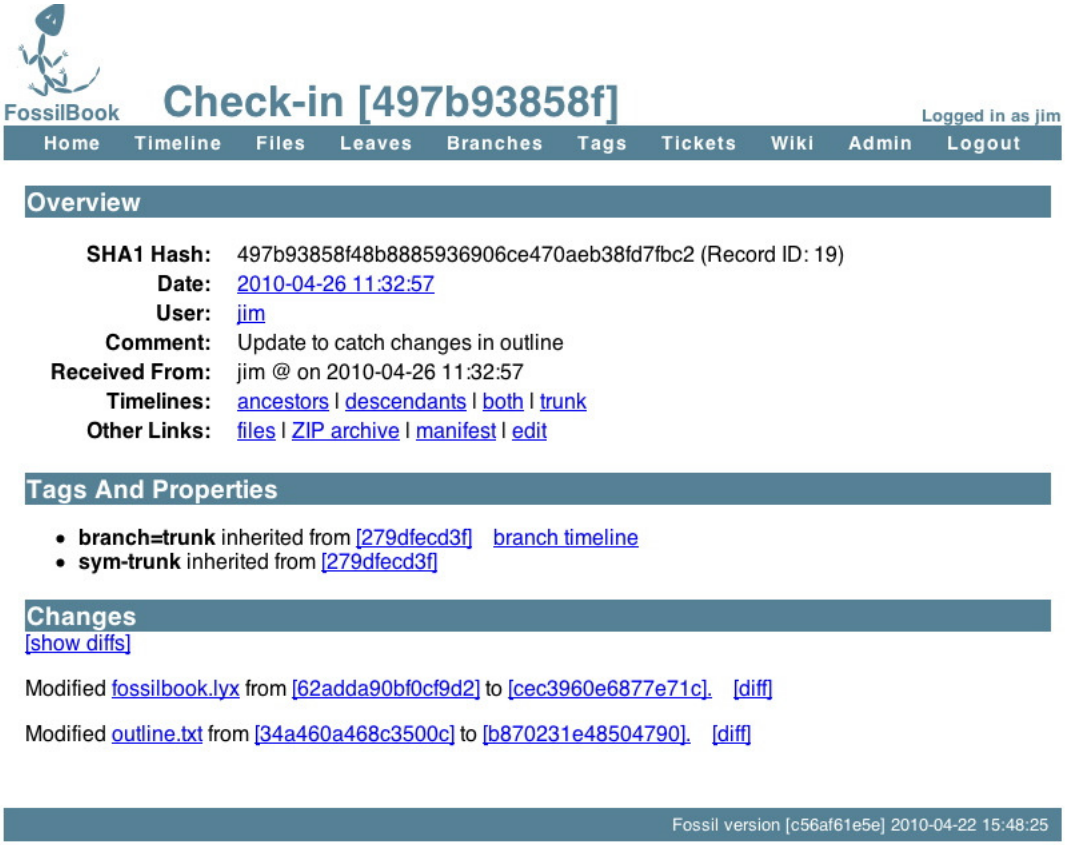

Figure 2.13.: 时间线细节

我点击了最新的检入操作的链接(那个**LEAF)** 后显示如上图。在这里你还可以做很多事情。从变更的文件清单你可以点击Diff链接查看那个特别文件所发生的变化。“其它链接”行里有一个非常有用的压缩存档。点击后将会从你的浏览器下载该版本的压缩文件。你会发现这在你想要获得某个特定版本时变得非常有用，事实上这个就是你用来从`http://www.fossil-scm.org/`获取最新版Fossil的普通方式。编辑连接在后面可以用来编辑一个Leaf。

2.4.1. 更新概要

- **fossil status**和**fossil extra**会告诉你没有提交前已经变更的文件和不在仓库里的文件
- **fossil commit - m “Commit comment”** 提交一个变更（或者多个变更），添加一个有含义的备注非常重要

2.5. Ticket-标签

除了管理你的代码之外Fossil还有一个问题追踪系统。这意味着你可以为一个问题或者你打算在你的系统中添加的新功能创建一个标签用来跟踪你的处理流程，当然你还可以绑定标签到你的特定的检入操作。对于软件来说修复BUG和增加功能时这个会非常有用。例如你可以在标签清单中查看BUG清单，然后注明这个BUG由你来修复它，这样你就很清楚你到底做什么，而不会被其它的变更搞得摸不着头脑。

当你点选了标签后会显示如下窗口，你可以新建一个标签，查看标签清单，或者生成一个新的报告。为什么保障简单，我现在开始只使用所有标签。

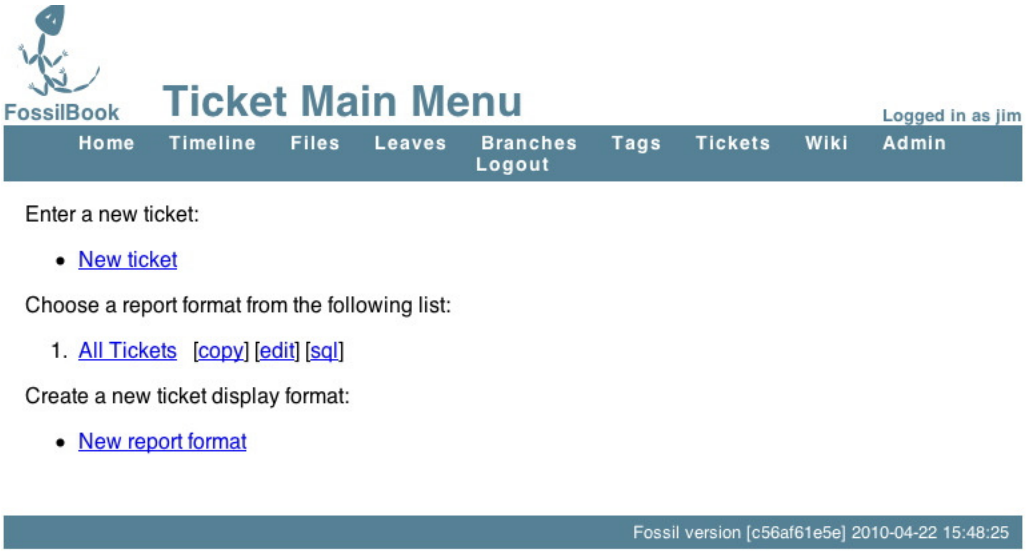

Figure 2.14.: 初始化标签窗口

点选“新标签”后我如下填写这个表单：

Figure 2.15.: 标签表单

事实上非常简单，这里你可以随意输入或多或少的内容，但是请记住这个东西至少会存在六周甚至是六个月之后，所以想象一下那时你会想要从这个表单里知道些什么（尽量多写写内容吧）。当我点击提交后如下图显示：

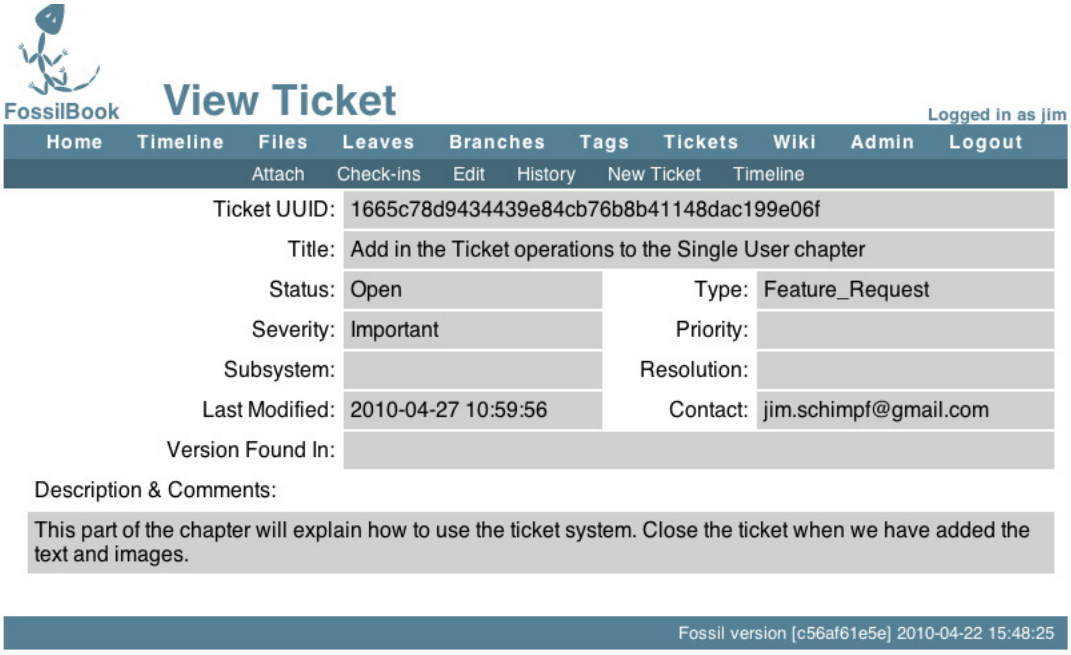

Figure 2.16.: 查看一个标签

最后点击标签然后“所有标签”就可以看到我的新标签以一个特别的颜色显示在标签清单里了。

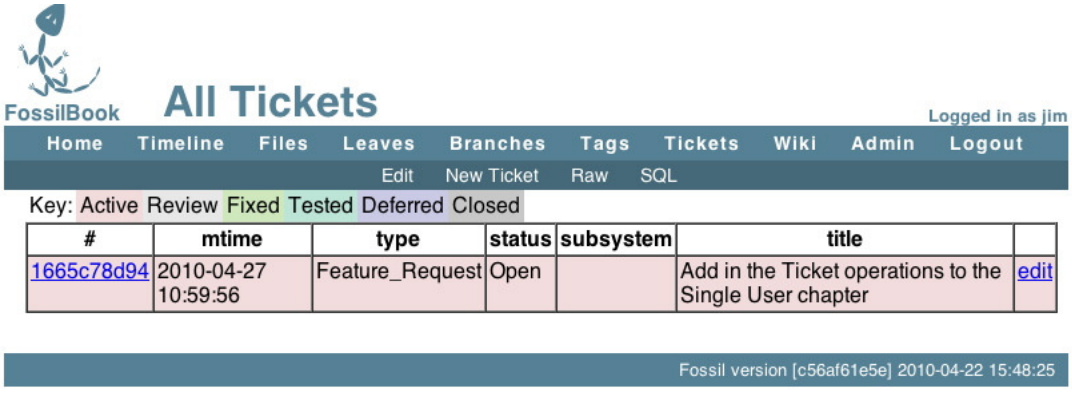

Figure 2.17.: 打开标签后的标签清单

我尝试，在处理标签时，使得标签指向或者链接到提交，且从提交可以回溯到标签。这个方式可以从标签查看以发生的变更或者从时间线我可以查看标签及其解决方案。为了达到此效果我将保证在使用Check-In（检入）的备注时时将从对应标签的十位哈希值输入进去并且在Ticket（标签）的解决问题处连接到对应的Check-In（检入）。

当我正在写这个章节并把所有的图片加入时，所要做的是增加所有的新图片到仓库并在另外一个完成的操作中进行检查。我所做的如下图所示:

```
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 2.18.: 检入解决问题的标签

第一步我将所有的新图片全部进行Add操作。我很懒，只是告诉程序将目录Single_user directory下的所有文件加入仓库。我之前已经增加过的那些如 **config_initial.epsf**但是Fossil很聪明，它知道不需要将它们添加第二次。尽管它们显示在已添加的条目里，但是实际上它们并未添加进去。

提交的那行命令非常关键，如你所见我在备注里方括号输入了标签的十位哈希值。我们将在Wiki章节看到这个是一个关联到Ticket标签的链接，所以当我们查看时间线内的备注时或者任何其它你可以点击相应括号跳转到对应的标签。

现在我已经将文件检入了，且到了不得不关闭标签的时候。我通过点击标签清单里的相应链接，页面将会跳转到标签窗口如图所示 2.17 on the preceding page。从那里我选择编辑并添加如下信息:

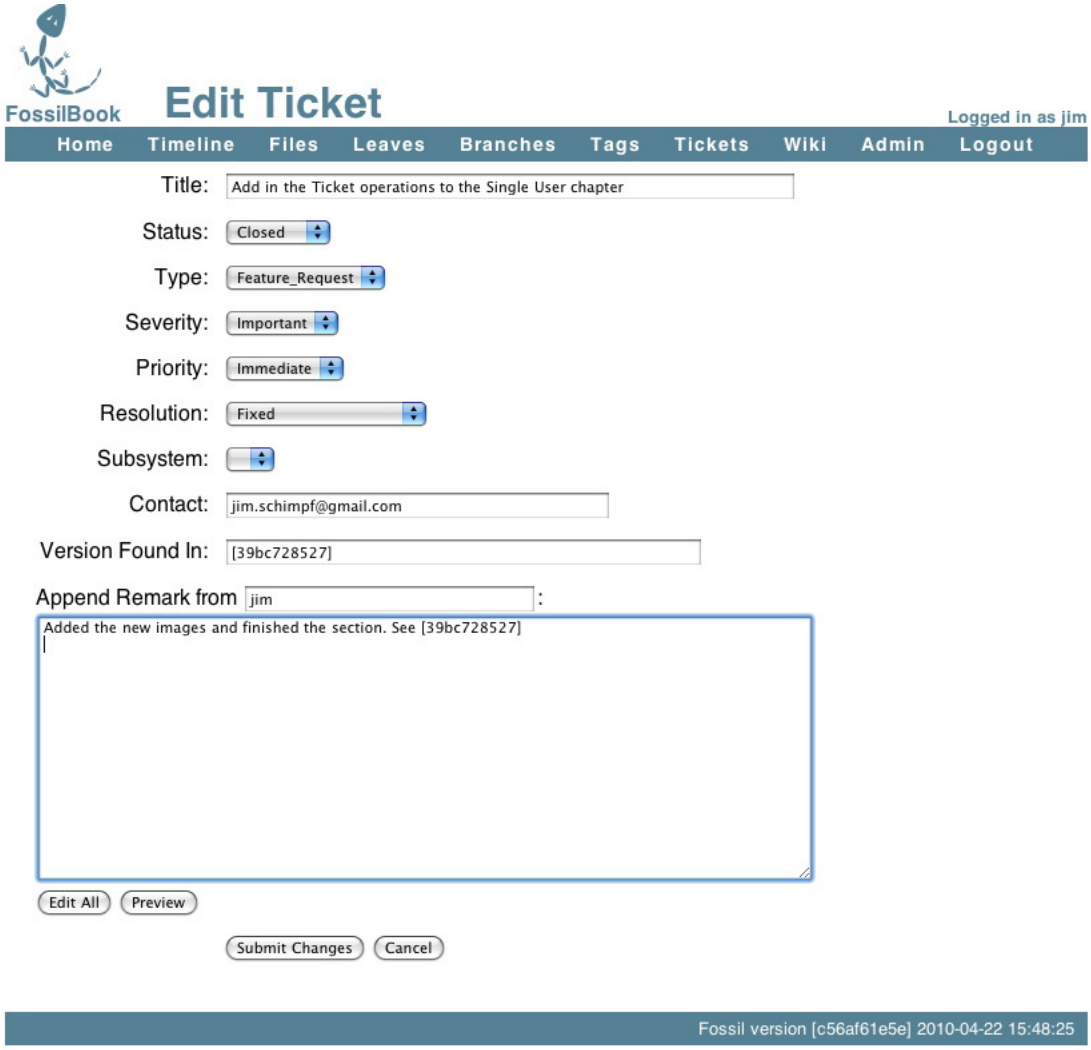

Figure 2.19.: 完成Ticket标签

我将它标记为“关闭”。如果你有相关状态代码你可以标记为修复、测试或者其它很多选择。另外一个非常重要的步骤，我提出了时间线并复制了提交（见： 2.18 on the preceding page）相关的链接。这样做了后我的标签现在可以链接到提交且提交也可以追溯到对应的标签。

2.5.1. Ticket-标签概要

- Ticket标签对于提醒你需要做什么或者修复BUG来说是一个非常有用的方式
- 当你提交一个变更时，将方括号括住的标签十位哈希值包含在提交的备注里比如[<10位哈希值>]，这样你就可以将其连接到对应的标签。
- 当你关闭标签时，输入哈希值以表示你已经完成对于的标签所描述的问题

2.6. Wiki的使用

如图解 2.5 位于 8页Fossil有一个基于浏览器的用户界面。该界面还附加的是，它内建了一个可以添加页面的Wiki系统。她允许你增加代码描述，用户手册或者其它文档。Fossil将它们保存在版本控制下的一个位置。一个Wiki是一个完整的网站，你可以在浏览器中添加页面和超链接。你有一个初始化的也面，然后你可以输入[newpage]，这个文本可以链接到一个地址且如果点击的话可以带你到一个新建的空白页面。记住图解 2.6 位于 8页中是你项目的一个初始页面，在这里你可以添加新页面。这些页面被Fossil自动纳入版本控制，而你不需要添加或者提交。

一旦我完成仓库的设置(见图解 2.7 位于 9页)，主页会变成如下图所示：

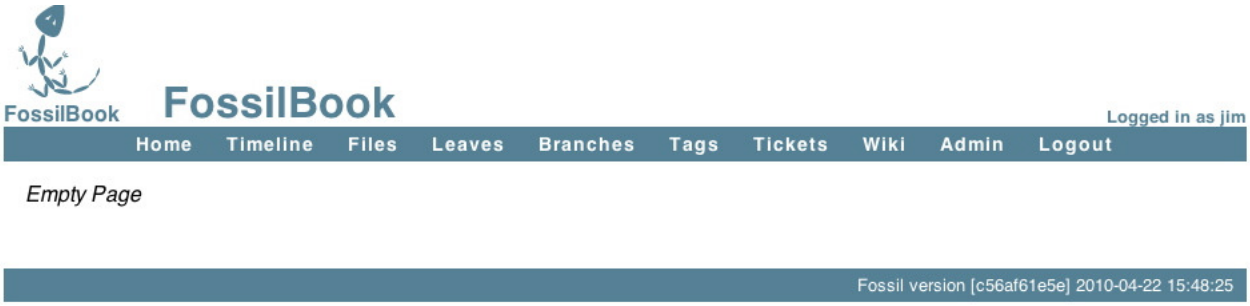

Figure 2.20.: 空白的主页

现在还不是非常有用吧，所以在这个章节的剩余部分我将使用Fossil的Wiki函数使得它变得有用起来。如果我从菜单栏处点选了Wiki的对象我会看到如下所示：

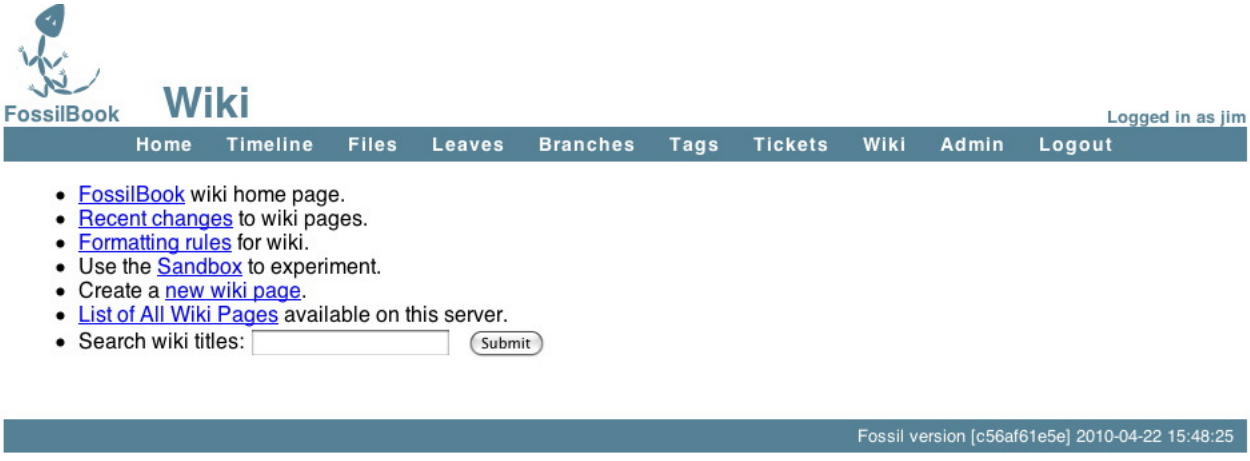

Figure 2.21.: Wiki控制

这些就是可以让你控制和修改Wiki的控制选项。目前来说最重要的是格式化规则链接。这个链接将带你到关于如何格式化你的Wiki版面的描述页面。如果你在页面里

仅输入文本它将会显示，但是会被你的浏览器格式化。你可以输入HTML命令来控制格式化。花点时间仔细阅读这个页面并了解什么可以做什么不可以做是非常有价值的。这个页面列出了有效的HTML命令，如果你不知道他们的意思，我建议你看看这个链接 http://www.webmonkey.com/2010/02/html_cheatsheet/并将它保存在你是手边。

除了非常有用的HTML标记以外，Wiki语法[page]非常有用，它可以链接到一个新的页面。这就是你如何添加新页面或者说是你如何为你的仓库添加基于网站的文档的方式。

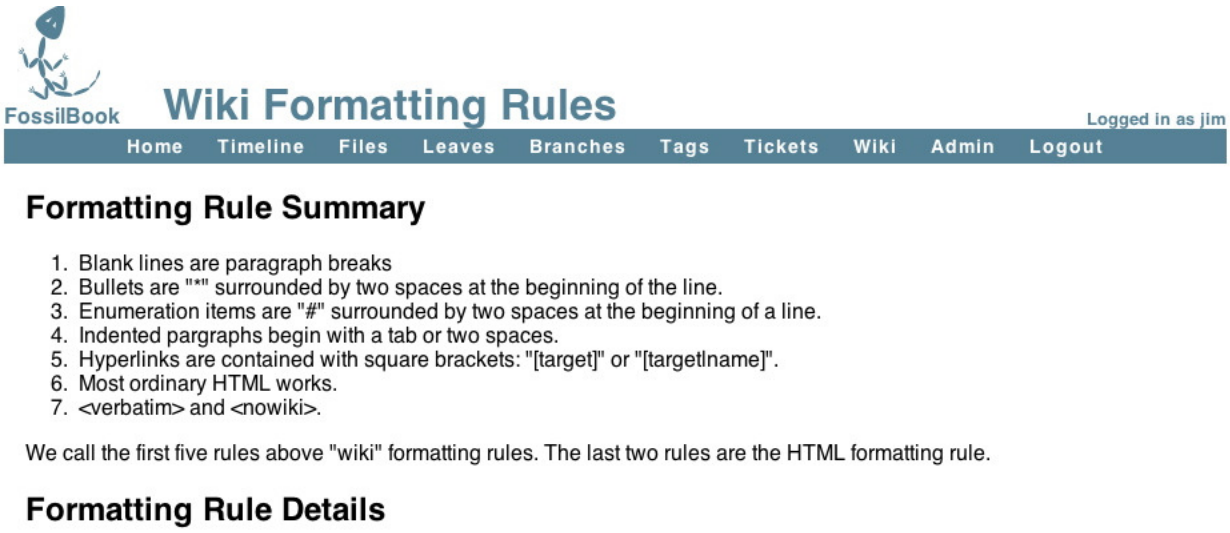

Figure 2.22.: Wiki格式化

我现在开始工作。我想要做的是修改首页并添加一个可以链接到本书PDF格式的超级链接。在图示 2.21 on the preceding page我点击第一个对象， FossilBook的主页。它又把我带到了主页但是现在多了一个编辑选项。同时还有一个历史选项，这样我就可以查看这个页面的历史版本了。

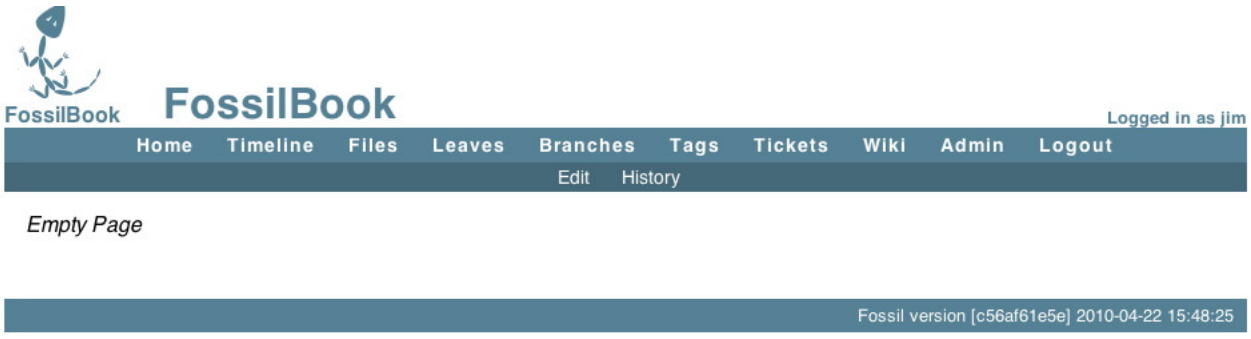

Figure 2.23.: 可编辑的主页

如下显示的是我的初始编辑及其预览:

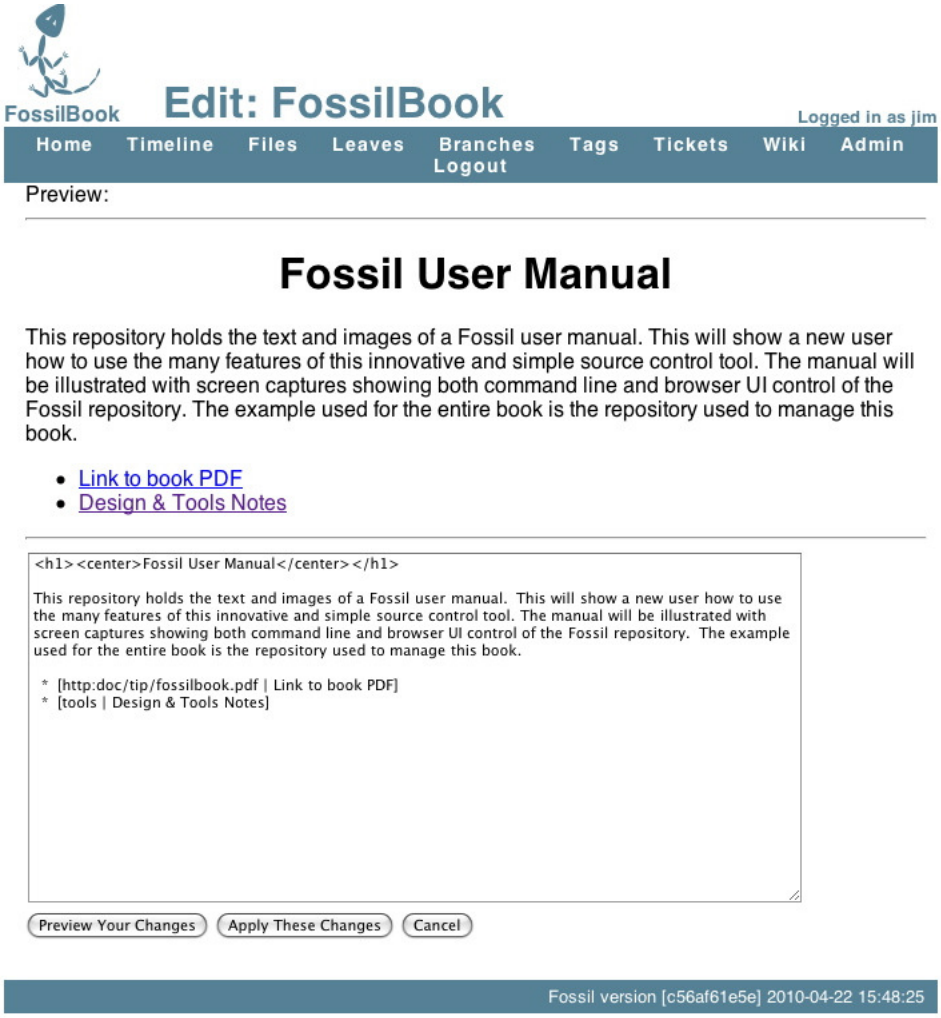

Figure 2.24.: 初始的主页

底部是一个编辑窗口，我可以在这里输入我想要显示的东西。上面的预览显示是这个页面将要显示的样子。如你所见我输入了一些简单的HTML标签来使得标题大一些并且居中。我刚刚输入的文体部分可以看到会自动适应到浏览器的窗口。你可以通过输入空白行来使用使用多个段落。下一步我想要一个列表，可以使用输入2个空格，一个'*'然后再是二个空格。我给每一行都添加到了一个新的页面（还没有添加的页面）。如果你看到我在表单里输入了[<new page> | <title>]。这样我可以用很长的字符来描述那个连接但是却有个短的页面名称(没有嵌入的字符串)。

在这里我经常犯的一个错误是点击新连接，它把我带到一个新的空白页面。**糟糕**，如果我还没有保存这个页面会发现我丢失了所有我之前的修改。

好吧，我将先保存我的修改然后再跳转到新的页面。这里我在做一些难以理解的事情，第一个链接连接到PDF，这是一个在仓库内的文件。LyX程序会创建这个PDF。我这么做，然后保存，并将它添加到仓库里。但是我不想链接到一个静态文件，而一直都是最新

版本的PDF，那个我每天完成工作的版本。要这样做可以如下文代码来实现:

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

这是一个特殊的链接，Fossil wiki可以理解**doc** 是表示这个是一个文档。**tip** 是意思是使用最新的当前版本；你可以添加一个版本号在这里。最后一旦我想要将书的PDF放顶部，我所需要的只是文件名称。如果它在一个子目录里，我可以这样写**doc/tip/subdir/filename**。

第二个链接只是常规的链接，我可以像我编辑主页的时候的方式来编辑它。

2.6.1. Wiki概要

- 使用HTML格式化你的文本,如<h1>标题</h1>用于页面头部
- 创建并链接页面，使用[<page> | <Link text>]
- 页面指定http:doc/tip/<仓库里的文档路径>将显示任何你浏览器可以处理的的文档（例如：pdf，http）
- 不要点击任何链接，除非你已经保存了当前页面

3. 多用户模式

3.1. 简介

在上一个章节中我只用了一个用户（我）来概述了Fossil的使用。在这节中我们将会学习在多个用户的情况下如何使用。感谢Fossil的分布式设计，一旦完成设置在多用户Fossil自动管理模式下的使用方式和单用户模式下没有多大的区别。

3.2. 设置

在上一节中Fossil仓库是一个我们操作系统里的一个文件，我们对这个文件进行提交代码和拉取源代码操作。Fossil是一个分布式的源码控制系统；这意味着存在一个所有用户都可以连接的主仓库。每一个用户在本地都拥有他们自己"cloned"的仓库副本，使用的命令和上一个章节**??**一样。现在不同的是Fossil将要把你的仓库和主仓库自动进行同步。

3.2.1. 服务器设置

我有一个叫FossilBook.fossil的仓库，现在要把它放在某个多个用户可以访问的地方。有二个方式，第一个是使用Fossil内建的网络服务器来托管这个文件，另外一个是使用操作系统支持的网站服务器和一个CGI类型的链接来托管（如果存在）。

3.2.1.0.1. 自托管 这是一个非常简单方式，下面就是可靠的保持机器存在且网络服务器已经启动。就这么多，当你完成一天的工作时不要关闭电脑，因为其他的用户可能需要使用它来上传代码，所有我要做的如下：

```
[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 3.1.: 自托管的Fossil仓库

这个是在UNIX系统里，在倒数第二行的"&"符号表示将Fossil的网络服务器在后台运行。如果你知道这台电脑有IP地址如192.168.1.200，那么在另外在同一个网络的机器上输入：

http://192.168.1.200:8081 浏览器打开这个链接，那么我就可以访问Fossil的网页界面了。

如你所见，这个很简单且可以运行在任何运行Fossil的操作系统中。你必须仔细确认它一直在运行且对于其它用户可访问，那么这就是一个非常简单的方式来保障主仓库的存在。

这个方法的问题是：

1. 如果你有多个仓库，你不得不使用**server**而不是 **ui**命令，且将所有的仓库都存放在一个目录下面。且它们都必须以后缀.fossil格式保存。
2. 如果电脑离线（比如:需要操作系统升级）或者其它原因无法自动重启Fossil服务器。
3. 仓库备份将无法实现。

这个方法可以工作，如果你只有一个仓库且主仓库的使用者足够勤奋努力，而且也会工作得很好。

3.2.1.0.2. 服务器托管 如果你有一个服务器类型的机器（比如：Linux或UNIX）可以运行Apache或者装有IIS的Windows机器，你可以使用它们作为你的仓库的网络服务器。这样有大量的优势：这个机器将会一直在线，而且会自动备份，并且可以很容易地实现多个Fossil仓库的托管。

我不打算深入如何设置网站服务器或者如何开启CGI，你可以查下如下网站

- Windows IIS 查看 http://www.boutell.com/newfaq/creating/iiscgihowto.html 只要保证你的fossil.exe在CGI脚本可以运行的路径里。
- Apache 查看这里 http://httpd.apache.org/docs/2.0/howto/cgi.html保证你知道CGI脚本保存在什么的地方。

如果你无法控制网站服务器，那么你需要服务器管理员的帮助开启CGI并将你的CGI脚本复制到正确的位置。

3.2.1.0.3. 托管服务器的CGI脚本 如果我们采用Apache服务器，就是我的方式，那么CGI目录位于 /Library/Webserver/CGI-Executables，然后我们需要写入如下的脚本：

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

Figure 3.2.: Fossil CGI脚本

然后把它放入CGI脚本目录。我将我的Fossil可执行文件放在目录/usr/local/binFossil仓库的共享的目录放在了/Users/Shared/FOSSIL。那么我的脚本应该作如下变化：

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

Figure 3.3.: My Fossil CGI脚本

在完成脚本后，我把它复制到了CGI目录并允许任何人允许它。

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

Figure 3.4.: 复制脚本到指定位置

3.2.2. 测试(含自托管和服务器托管)

如果所有文件都在正确位置那么我现在应该可以访问网站服务器且他会将我带到如下界面：

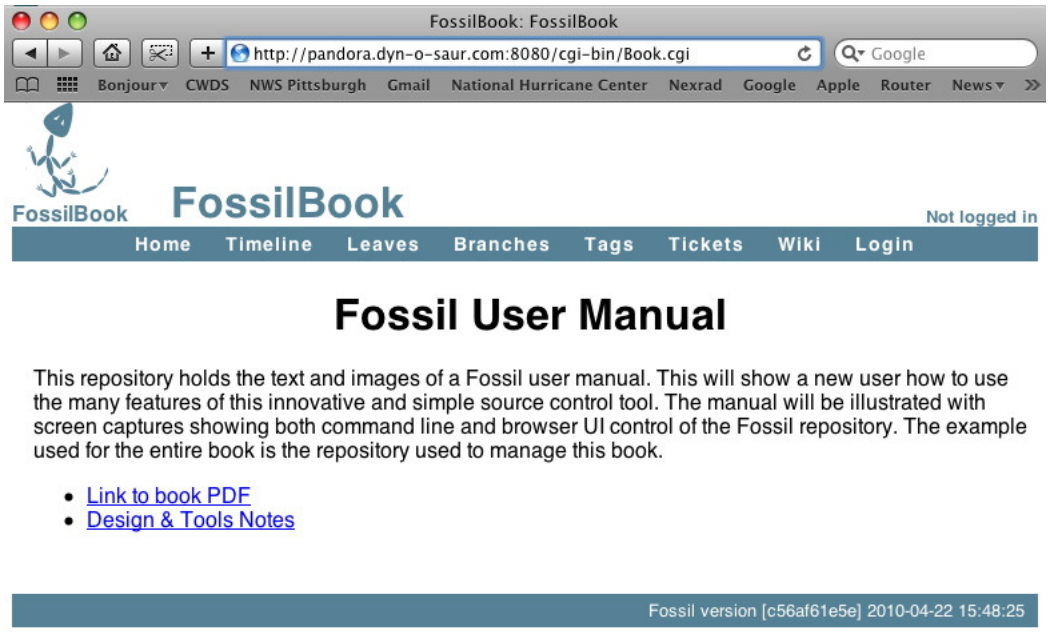

Figure 3.5.: 网络访问Fossil CGI托管的网站

3.3. 用户账户

为一个仓库提供网络服务，不管是自托管还是复杂的CGI方式，都会将你带到同一

个地方，如图示 3.5 on the previous page所见。现在我需要设置用户账户以方便其它贡献者为我的书提交代码或者修改。记住Fossil已经自动创建了一个匿名用户（见图示 2.8 位于 9页）。其它人可以在有限制的情况下访问这个网站，默认匿名用户可以下载这本书但是不能提交修改。在这个项目组我打算创建一个新账户（Marilyn）且这个账户可以修改及提交修改，因为她是我的编辑。

为了完成所有这些操作我需要在登陆界面输入我的ID(Jim)及我的密码。现在我是超级用户，我可以回到用户配置页面，图示(2.8)然后添加一个新用户：

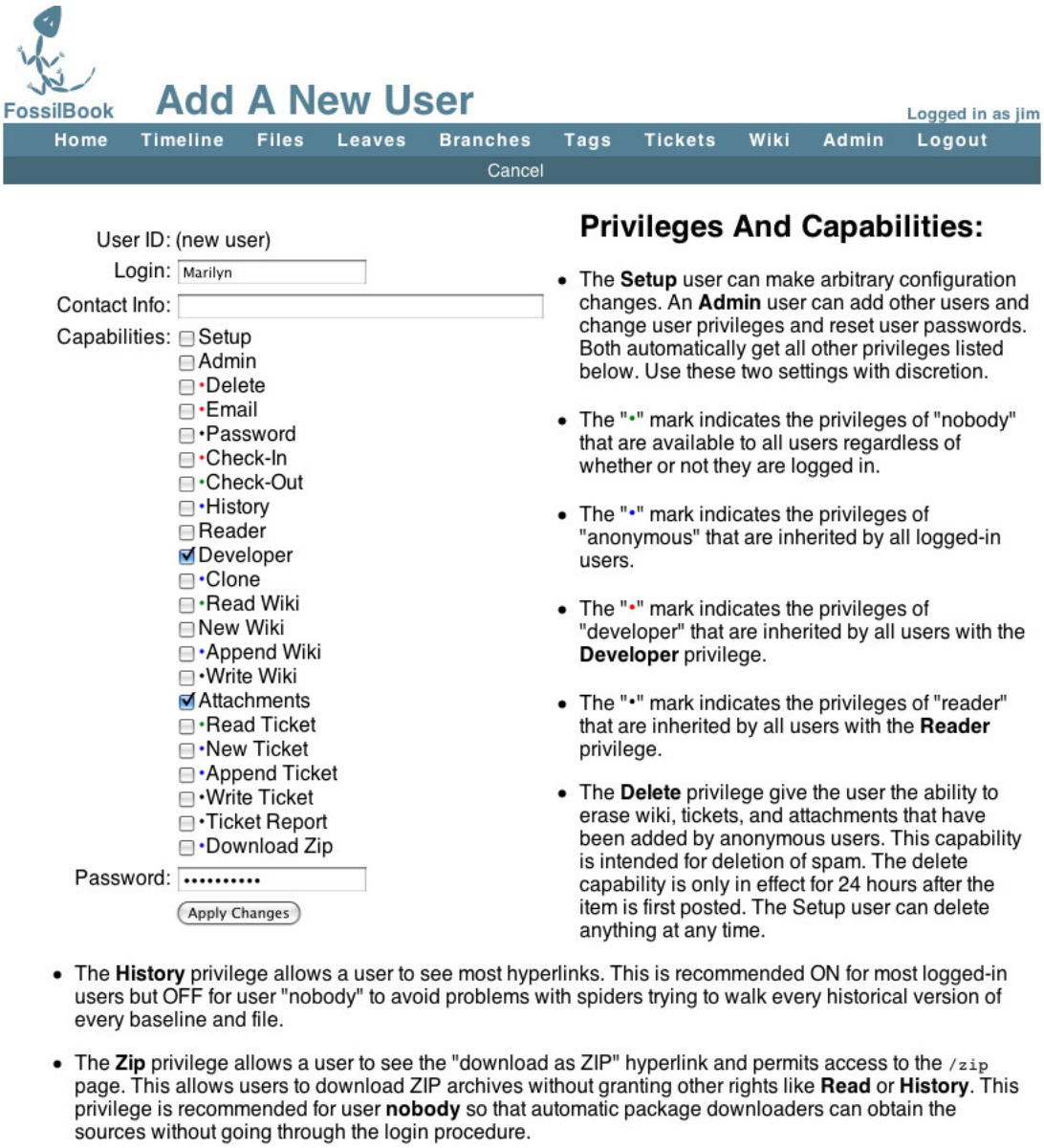

Figure 3.6.: 新编辑用户

我打算把她设置为一个编辑，如果我们需要编写代码那么需要她作为一个类似开发者的角色，所以我为其选择了开发人员特权的级别。这样的话她就可以对仓库执行check-in、check-out和写和更新标签的操作。我还添加了添加附件的权限，因为考虑到她需要做些如放图片及其它评论类的操作。我还给她设置了一个密码，那么她的连接将会被密码保护。

在这里，我还可以添加其它的用户但是该项目不要其它的人员了，但是你可以看到这个操作是多么的简单了。当你指派给用户权限的时候只要仔细阅读并保证不要给他们你认为不需要的功能。如果他们有问题，你可以在将来给他们做简单的修改就可以了。

3.4. 多用户运行

当服务器已经设置完成并在运行，下一步用户确定需要做的就是Clone这个仓库。这个操作的就是从网站服务器将仓库复制到我的本地机器。一旦完成Clone操作后，使用这个仓库的方式和单用户使用模式 **??** 位于 ??页下的操作命令几乎一模一样。Fossil将同步你的本地仓库到服务器。

3.4.1. 克隆

在Clone一个Fossil仓库之前，你需要知道下面四个事情:

1. 网站地址，我们目前的仓库地址是 **http://pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi**
2. 你的账户名称，我的例子中是**jim**
3. 你的密码（我还是不会告诉你的）
4. 本地仓库的名称，在本书的例子里叫Fossilbook.fossil

然后你可以在任何你想要保存仓库的目录（本书例子的目录是FOSSIL）里运行如下克隆命令

```
[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 3.7.: Clone命令

在这里我可以参考章节 **??** 位于 ??页的步骤来设置我的用户密码，然后将仓库在一个工作文件夹内打开。

现在我已经把所有的东西转移到了新克隆的仓库里，我在完成一天工作后的检入操作如下所示：

```
[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 3.8.: 克隆仓库的检入

如你所见文件被提交到了本地仓库然后本地仓库自动与服务器进行了同步。

3.4.2. 保持同步

在完成了所有上文所描述的设置工作后，我现在拥有了一个分布式的源码控制系统。我的合作者Marilyn也已经克隆仓库并开始工作了。她在编辑病修正书中我语句不通顺或者有拼写有误的地方。她独立地工作在同一个且同时被我在使用的文件。我们必须使用Fossil来阻止我们同时编辑FossilBook.lyx文件，却是不同的编辑方向（译注：一个在创作，一个在修改）。记住Fossil没有锁定任何文件，没有任何东西阻止她编辑或修改当前我正在工作的文件。

这里我们都必须遵循流程以阻止这个问题的发生。尽管我看不到她所编辑的文件的修改，直到我们都将修改提交主仓库。在我已经提交了且她的使用的是当前最新版本时,同一文件拥有二个版本并不是问题。

这里有二个问题：

1. 在我开始任何工作前我必须保证我拥有所有文件的当前最新版本
2. 当我提交时我必须保证我的修改部分不是她已经修改的部分

第一个问题是非常明显的，在你工作前保证你有最新版本。我们通过使用Update命令来实现更新。见图示3.8，我完成了最新的检入操作。在开始任何工作前我应该保证Marilyn没有检入其它的文件。我可以检查时间线但是我将替代使用更新操作更新我的仓库和源码文件。当我更新时，我指定它需要从**trunk**更新。这个保证了我是从最新的仓库更新而不是其它什么分支。

```
[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 3.9.: 更新动作

哈哈！Marilyn已经在工作且更新了源码和PDF。如果我从网站服务器检查时间线查看，可以知道她甚至写了相关的文档:

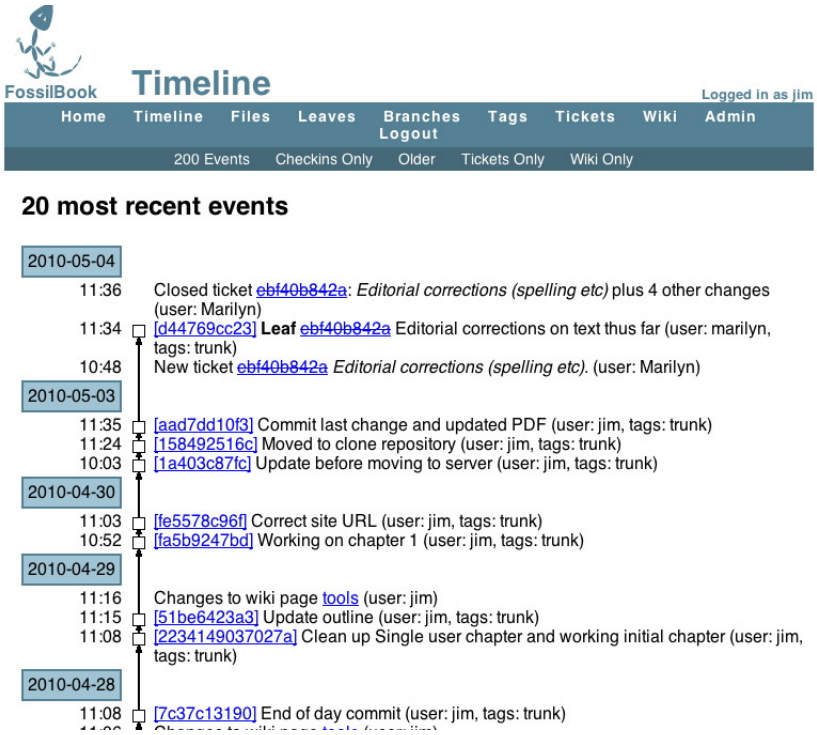

Figure 3.10.: 更新后的时间线

现在我知道我有最新的状态了，我可以继续增加新章节的步骤了。

3.4.3. 复杂化

在章节 3.4.2 on the preceding page的第二个案例比较难。在这个案例里我开始工作前很勤快地进行了Fossil更新，同时Marilyn也完成了她的更新并开始了她的工作。现在她完成了

并在对她的修改进行检查。很显然我不知道这些，因为我在高兴地进行我的修改工作。下面会发生什么... ...

```
[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 3.11.: 作为分支提交

哈哈，恰好的事情发生了，Fossil警告我说我的文件副本与主仓库副本不一样。如果我添加强制提交的命令–force的话，仓库将会生成一个分支而Marilyn将来提交时会被提交到她的分支且我的提交只会提交到我的分支。那样不是我们想要的因为我想让她编辑 我的副本。

下一步按照Fossil说的需要更新我的副本。这样的话你必须得小心了，因为盲目地更新那些修改过的文件会覆盖我刚刚完成的东西。所以我们通过添加-n和-v选项参数来执行一个尝试更新，“执行一个不发行的更新”并且给我看看结果。

```
[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 3.12.: 不发行的更新

如你所见，几乎所有的东西都是UNCHANGED，我要做的不多除了fossilbook.lyx需要合并（ MERGE）及fossilbook.pdf需要更新（UPDATE）。这就是我所期望的，Marilyn对文件fossilbook.lyx完成了她的修改工作，所有我们必须要合并她的修改。但是她还更新了fossilbook.pdf而我没有这个版本。在我们继续工作之前，如果你在使用Linux或者UNIX系统，你可以如下执行一个不发行的动作：

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

Figure 3.13.: 小范围的不发行更新

通过使用管道和grep命令，我可以消除所有没有变更的条目。

3.4.4. 修复更新文件

首先我们修复比较简单的文件，对于fossilbook.pdf我可以直接更新它到仓库当前版本以实现与其匹配。它不需要被合并所以只要替换它就可以了。在我做这一步之前我需要查看时间线。

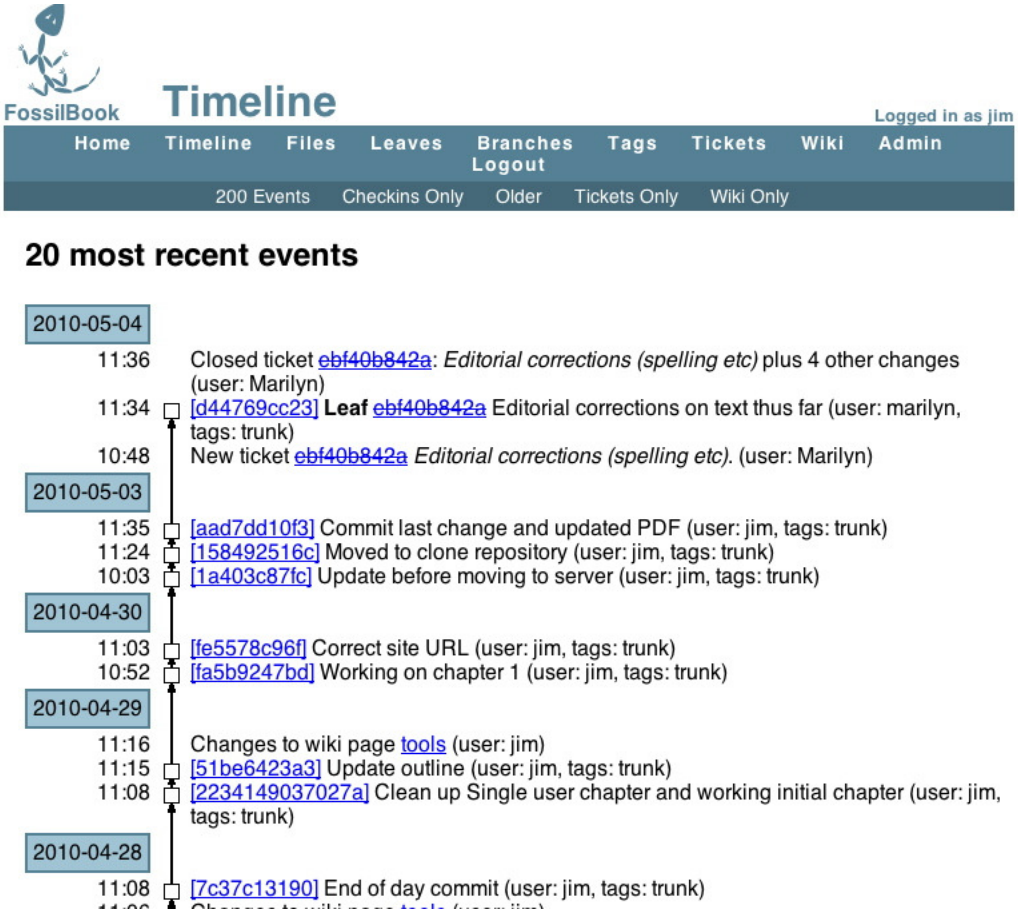

Figure 3.14.: 当前的时间线

我看到当前的 **Leaf**（片段）是[d44769cc23]且它被标识为 **trunk**。我想要从它那里更新fossilbook.pdf这个文件，那么我输入：

```
[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 3.15.: 更新fossilbook.pdf

好了，它完成了更新。

3.4.5. 修复合并文件

我们可以使用Fossil内建的工具。这当前这个案例提示如果提交的话将产生一个分支Jim将会使用–force选项来创建分支并且稍后处理合并的问题。

```
E:\Profile\Ratte\data\organize\fossil-w32\fossil-book>fossil commit -m "adding some changes of jim"
fossil: would fork.0 "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 3.16.: 命令窗口的强制提交

现在的时间线如下图所示：

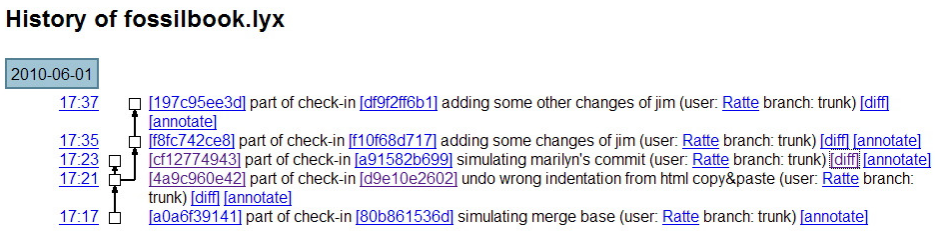

Figure 3.17.: 已分支的时间线

我们使用Fossil的合并命令来删除这个分支（如：将Marilyn的变更内容纳入到trunk里）。我们可以使用merge是因为fossilbook.lyx是一个文本文件，而Merge命令的作者将其设计为可以在文本文件上进行工作的命令。如果它是一个二进制的文件我们必须使用一个外部程序或者使用对应文件格式的处理程序来复制黏贴来合并二进制文件。

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

Figure 3.18.: Fossil合并

在文本编辑器里查看文件(fossilbook.lyx)(不是LyX)我们发现：

```
>>>>>>> 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 3.19.: 文本的不同之处

在完成提交之后，时间线告诉我们如何将分支合并到了主分支。Marilyn将需要更新到这个心的分支。(见章节 **??** 位于 ??页)

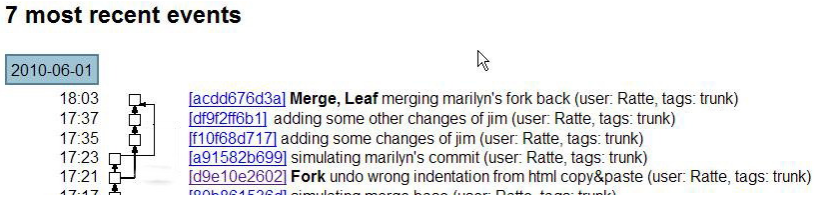

Figure 3.20.: 合并的时间线

4. 分叉 & 分支

4.1. 简介

这个章节将会覆盖Fossil的分叉和分支。 分叉是你非故意创建的二个检入仓库的不同源码版本。分支(Branching)是你刻意创建的，因为你想要在同一个仓库里维护二个或者更多的源码版本。我在章节 3.4.3 位于 29页里提到了分叉（Forking）及其使用方案。如果，不修复（合并）直接提交，且强制提交后，Fossil将会自动在仓库里创建一个分叉（Forking）。

分支（Forking）通过创个二个检入（checkin）的路径来避免冲突，因此不同的用户将会在不同的路径上工作且可以检入相互矛盾的变更。而分支（Branch）却是你所期待的另一个选择。当你想要拥有二个基于不同源码的不同版本同时进行工作时可以使用分支（Branching）。如在章节 1.1 位于 1页所描述的，当你期望在同一个仓库里有一个生产版本用于维护一个开发版本用于生产，这样的话生产的变更只会被检入到开发分支（Branching)，维护的修改可能都会被检入到上述二个版本。

这里我们不用这本的仓库作为示例，而是用有大量文件盒文档的JSON[1]语法程序作为例子。这样来阐述Branching和Tagging会变得简单一些。

在Fossil网站上有一个比较好的帖子可以参考http://www.fossil-scm.org/index.html/doc/tip/www/branching.wiki。

4.2. 分叉，分支& 合并

在这个例子里，JSON代码已经添加到了Fossil且有二个开发人员已经检出了代码并且开始了他们各自的工作。Jim想要修复修复大量的编译警告而Marilyn想要修改文档。他们所做的操作流程乐意参考章节 3 位于 23页。JSON 的源代码分布在不同的位置，他们都克隆（clone）了仓库，且打开了代码的工作拷贝。

4.2.1. Marilyn的操作

她检阅了文档并发现大量的问题并修正了这些问题（文档使用的是LyX和PDF格式）。当她对她所完成的时间感觉很满意后，她将她的修正版本检入到了如下所示的地方：

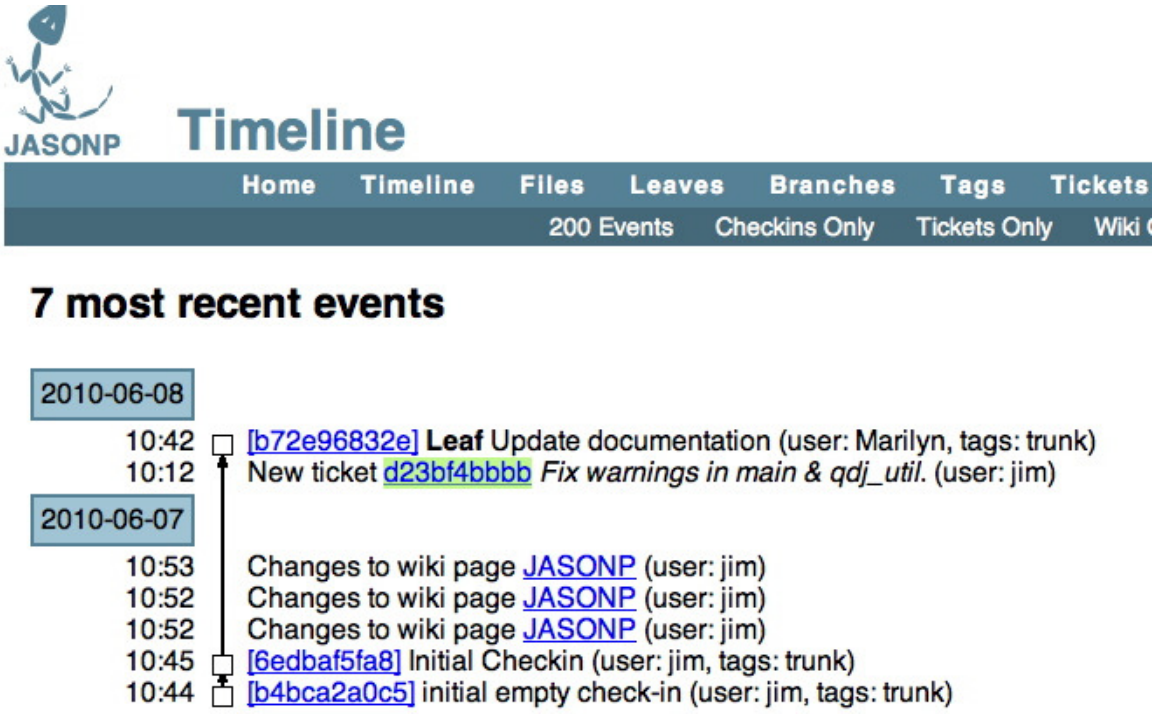

Figure 4.1.: Marilyn的工作

4.2.2. Jim的操作

同时，Jim获取了版本[6edbaf5fa8]的源码副本，并如4.1章节加入了一个标签（ticket）[d23bf4bbbb]。在完成了修复警告的问题后，Jim完成并进行了提交。他做这些是在Marilyn完成她的提交之后。

```
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 4.2.: Jim的提交意图

这时Fossil意识到Marilyn已经修改了仓库（她更新了文档）但是Jim却没有这些修订的内容，因为Jim检入的是一个更早版本的源码。Jim认为他**必须**获取更新，所以他使用FORCE参数强制进行了提交。

```
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 4.3.: 强制的提交

看看时间线，Jim看到如下内容：

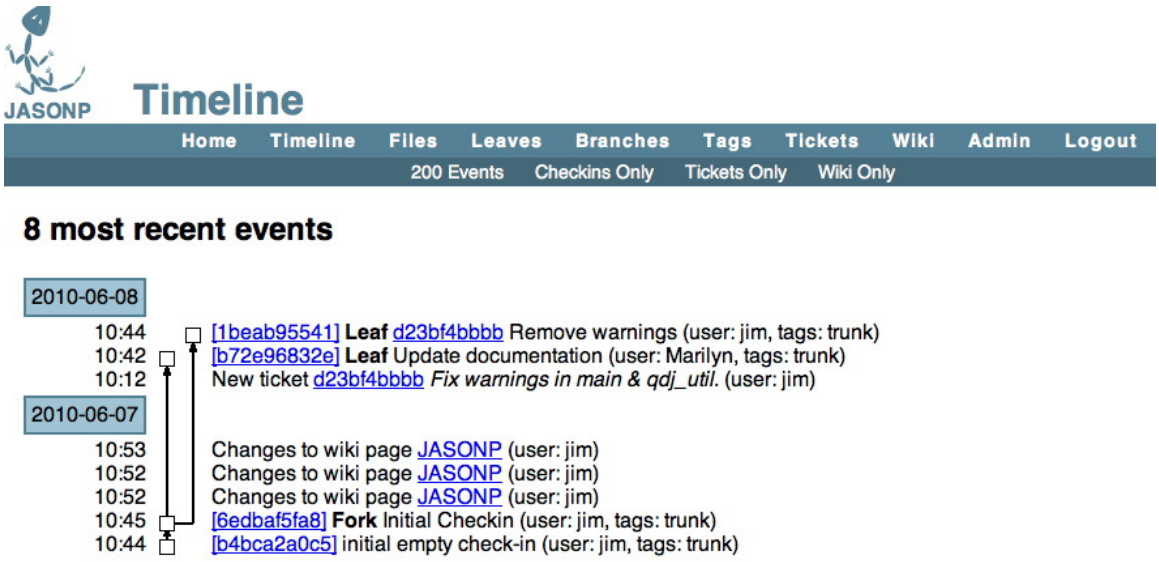

Figure 4.4.: 仓库分叉Fork

不好，现在有两个**Leaf**， Marilyn可以将她的修改提交到她的Leaf，Jim也可以将修改提交到他自己的leaf。所以Jim必须通过合并来修复这个问题。Jim想将Marilyn的版本和他的版本[1beab85441]进行合并。

4.2.3. 修复分叉fork

所以Jim从检出了分支Leaf[1beab85441]并且与Marilyn的分支leaf[b72e96832e]进行合并，操作如下：

```
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 4.5.: 合并操作

如图有二个文档文件已经被更新，而且没有任何需要合并的冲突发生，因为Jim没有动任何文档的文件，Marilyn也没有动任何程序代码文件。

下一步Jim进行了提交，将这个新合并的那些文件添加到了主分支。记住图示4.5的合并只是更新了你检出的代码而没有将其提交到仓库，除非你将它们检入（主仓库）。

```
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 4.6.: 合并后的提交

当我们查看时间线时我们只有一个Leaf，将来可以向这个Leaf提交代码。

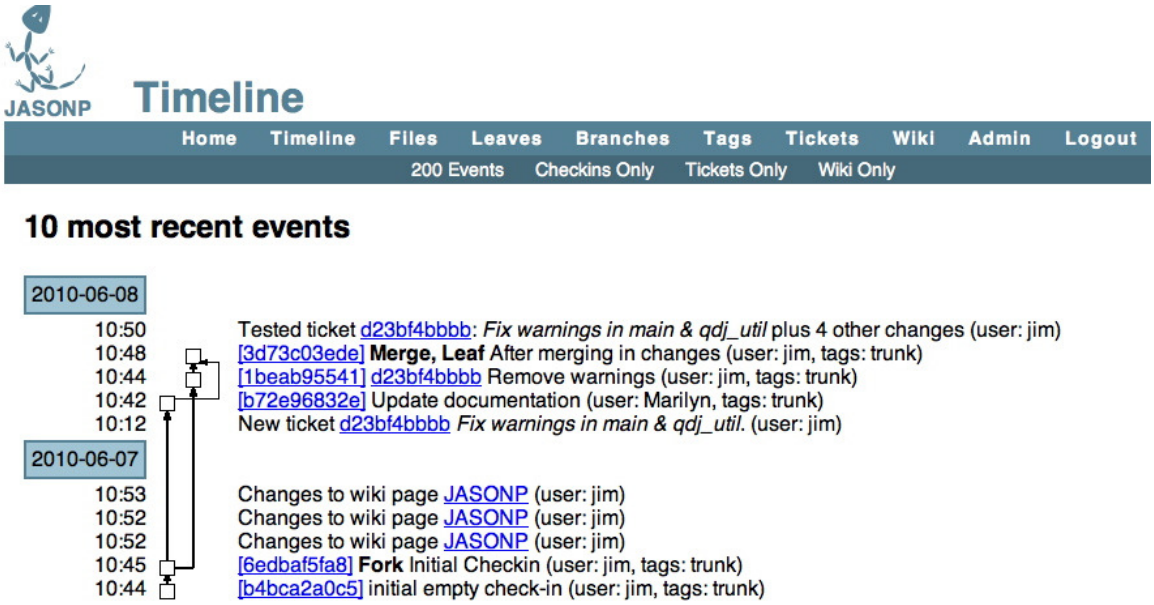

Figure 4.7.: 合并后的时间线

还剩余一件事情，就是Marilyn需要在继续工作前进行更新，所以她为了与主仓库同步需要检出最新的代码。

```
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 4.8.: Marilyn的更新

4.2.4. 使用的命令

- **fossil merge <fork>** 用来合并一个分叉（通过哈希值来指定分叉）到当前的检出版本
- **fossil update <version>** 如果不是使用默认标识的当前版本的检出时，用来更新当前检出到指定的版本（见fossil status）

4.3. 不合并的分叉

在这例子里，我将演示如何在多用户情况下合并代码的修改而不会产生分叉。例中Marilyn加入了一个BSD许可证的文本内容到所有的代码文件中而Jim正在为代码增加一个帮助函数。例中他们都写了一个标签来说明他们做了什么，但是他们各自独立行事。

4.3.1. 检入（Checkin）企图

Marilyn第一个完成并检入了她的修改。Jim编译、测试并检入代码却看到如下信息:

```
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 4.9.: Jim的检入企图

4.3.2. 更新

下一步Jim的操作是执行更新但是没有修改，使用-n参数来显示如果不更改任何文件修改将要发生什么事情。

```
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 4.10.: 更新演练

这里显示一些文件将会被更新，如被仓库里的新文件所替换。就是说将会混合Jim修改的和仓库里的文件内容。注意使用参数**MERGE**的意思是二个文件的合并是分离的。也就是说合并在没有人为介入自动完成。

Jim真正地进行了更新然后提交合并后的文件，从而新建了一个leaf。那么现在我们将Marilyn和Jim的修改合并到了最新的版本里。

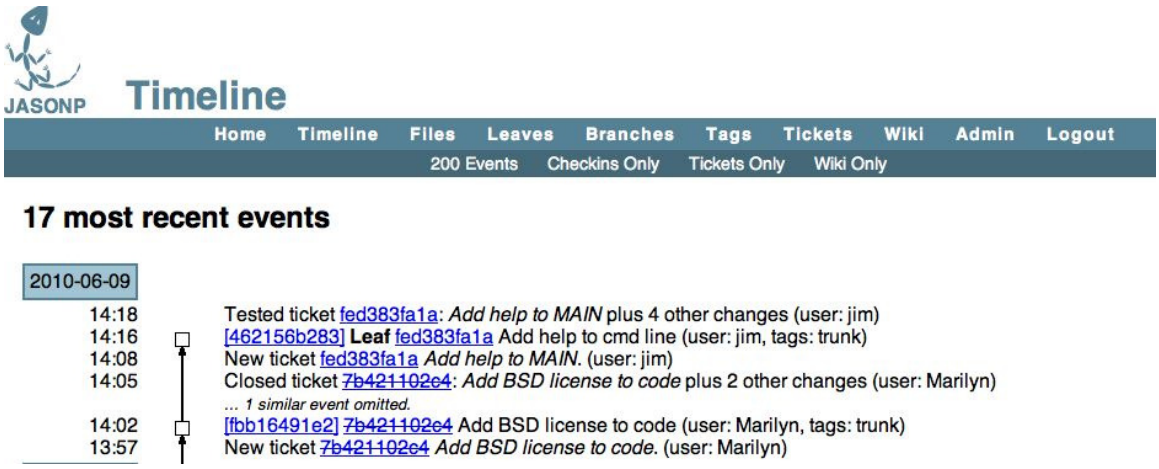

Figure 4.11.: 合并后的仓库

4.3.3. 使用的命令

- **fossil update -n** 执行一个更新的演练以查看什么文件会被更改。
  - UPDATE 意味着文件将会被仓库的文件替换
  - MERGE 意味着将会将仓库的文件和已检出的文件进行混合

4.4. 分支Branch

4.4.1. 简介

我们之前已经讨论过这个，但是分支是故意将仓库里的代码拆分到不同的路径。这个一般发生在我们有维护分支和开发分支的情况下将生产代码进行区别。维护分支已经被使用并会基于体验进行BUG修复；开发分支获取这些修改并合适地修改以增加新功能。

JSON代码语法解析器已经被测试和使用，并已经发布应用。我们想要修改它使得其可以支持UTF-8编码的字符，这样它才是一个可以匹配JSON的标准的程序。当前的版本只能工作在ASCII七位字符下，是一个不标准的JSON库。我们打算将代码拆分为VER_1.0分支，也就是当前正在使用的代码；另一个是VER_2.0分支，将会被用来增加UTF-8编码的支持。

4.4.2. 将仓库添加分支

在操作前我们需要确认我们检出的代码和主分支trunk一致。

```
[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 4.12.: 检查代码状态

看到它与时间线上最新的leaf是匹配的，我们可以继续对代码进行分支操作了。

```
[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 4.13.: 分支命令

刚才做了什么？我们使用Fossil的分支命令创建了二个分支VER_1.0和VER_2.0并给它们各自指定了一个颜色，我们现在可以查看时间了：

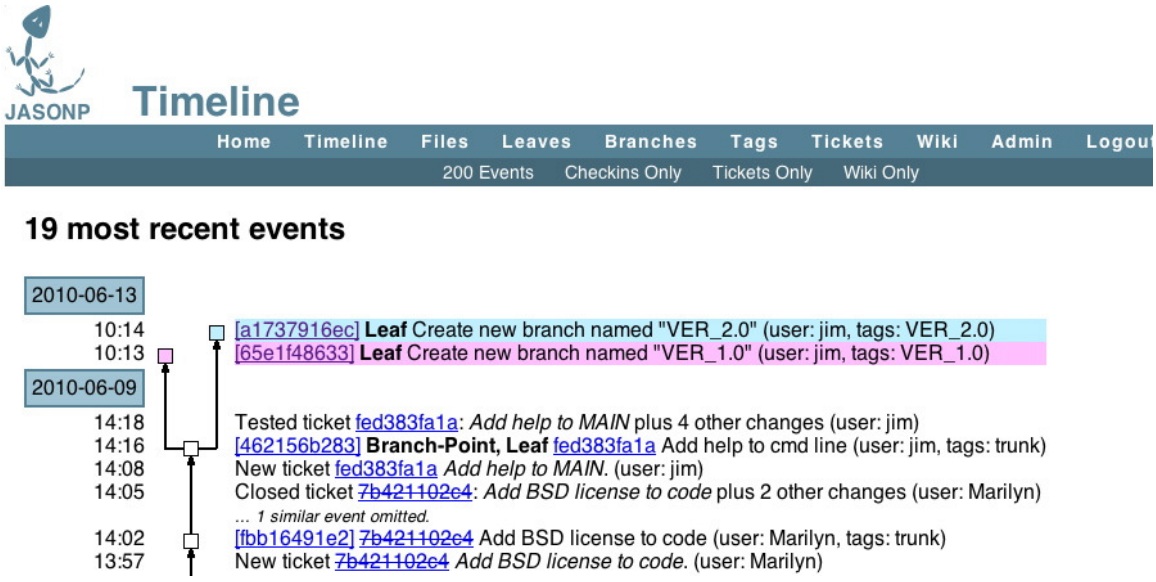

Figure 4.14.: 分支时间线

4.4.3. 颜色设置

如你所见二个分支在时间线上有不同的颜色。这是在我们创建分支时由**-bgcolor**选项增

加的。(见图4.13)。但是我们想让这个颜色可以显示在每个分支以后的检入操作中，要达到这个效果我们必须使用UI命令并为时间线选取特定的Leaf。

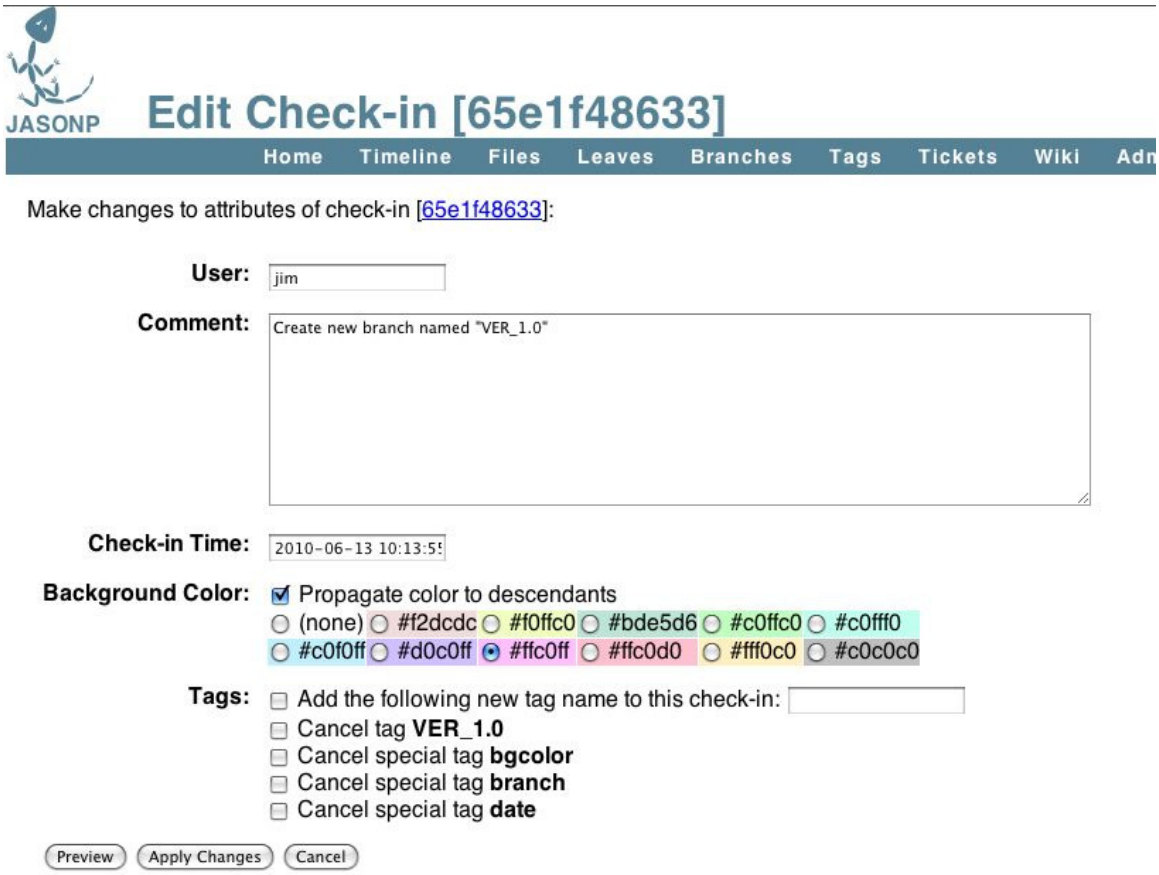

Figure 4.15.: 设置分支在时间线上的颜色

在章节**背景颜色**下我选择了**传递颜色到子对象**，那么将来的检入操作将会使用相同的颜色。

4.4.4. 检出分支

现在仓库已经有分支，我们就可以在不同的文件夹内检入不同的分支。我们创建了jsonp1和jsonp2文件夹并执行打开命令将不同的分支释放到这二个文件夹内。

```
[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 4.16.: 检出VER_1.0

同样的方式检出VER_2.0

```
[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 4.17.: VER_2.0检出

注意Tags标签，这样我们可以知道我们在那个分支上。

4.4.5. 修复错误(全部)

在完成这些工作后，我发现文件main.c有一个关于未使用的变量的警告。我想在二个分支中都修正它。目前二个分支中的文件都是一样的，所以在每个分支修改提交或者覆盖到另外一个再提交是可行的。我新建了一个关于main.c修改和编辑的标签。我把它复制到每一个分支然后分别检入。

```
[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 4.18.: 修复二个分支的警告

现在的时间线如下所示：

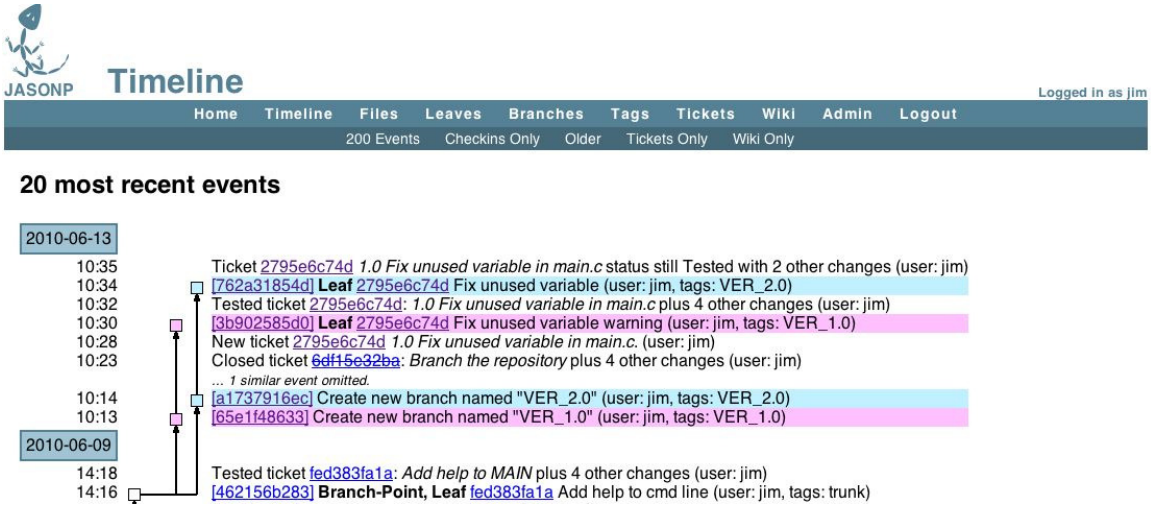

Figure 4.19.: 修正二个分支

4.4.6. 使用的命令

- **fossil branch** 用来生成一个仓库的分支。这个命令有给分支指派颜色的参数选项。

5. Fossil命令

5.1. 简介

这个章节将讲述Fossil命令行命令。这部分将会分为多个章节，第一部分将讲述必须知道的命令。这些命令是你一直都需要使用的并且需要记住缩写。其它的命令将会分为维护、高级和杂项。这些命令只需要你在使用它们之前查看参考即可。

最重要的命令是**help**，你可以在命令行输入**fossil help**程序将会列表显示所有已有的命令 。然后输入**fossil help <具体命令>** 将会在窗口打印出关于这个命令的详细信息，你会一直可以使用这个作为参考。本书的这个章节将会尝试通过添加一些例子和深入的解释来补充内建的帮助。所有的命令将会建立索引已方便进行搜索查找。

注意： Fossil是一个活动的项目，命令可能会增加也可能会在将来的版本中删除。输入 **fossil help** 以获取你当前版本的最新命令清单。下面的显示的是我写书时的fossil命令你使用的版本可能会有些不一样。

5.2. 基本命令

5.2.1. help

这个命令来输出当前Fossil版本的所有命令。它还可以使用格式**fossil help <command>** 来获取任何命令的详细信息。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help
Usage: fossil help COMMAND
Common COMMANDs:  (use "fossil help --all" for a complete list)
add         changes     finfo       merge       revert      tag
addremove   clean       gdiff       mv          rm          timeline
all         clone       help        open        settings    ui
annotate    commit      import      pull        sqlite3     undo
bisect      diff        info        push        stash       update
branch      export      init        rebuild     status      version
cat         extras      ls          remote-url  sync
This is fossil version 1.25 [d2e07756d9] 2013-02-16 00:04:35 UTC
```

Figure 5.1.: 运行Help命令

事实上，这个命令只将会列出命令帮助的子集，限于经常使用的命令。如果你想查看所有可用的命令可以使用**fossil help –all**命令来处理。不同的版本之间你可以看到不同版本所包含的帮助子集的变化。

一个使用**help**命令来查看特定命令的进一步详细信息的例子如下：

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help add
Usage: fossil add FILE...
Make arrangements to add one or more files to the current checkout
at the next commit.
When adding files recursively, filenames that begin with "." are
excluded by default. To include such files, add the "--dotfiles"
option to the command-line.
```

Figure 5.2.: 具体命令的帮助

5.2.2. add

增加命令用于将文件加入到仓库里。它是递归的会将所有当前目录下及子文件夹内的文件都推入仓库。Fossil不会覆盖任何已经在仓库里的文件，所以任何时候添加所有文件是安全的。只会将添加新文件。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help add
Usage: fossil add FILE...
Make arrangements to add one or more files to the current checkout
at the next commit.
When adding files recursively, filenames that begin with "." are
excluded by default. To include such files, add the "--dotfiles"
option to the command-line.
```

Figure 5.3.: add命令

输入：

```
fossil add .
```

将会添加当前目录及子目录的所有文件。

注意所有这些文件在提交前都不会被推入仓库里。

5.2.3. rm or del

命令rm用于从仓库删除文件。但是文件在你下一次提交后从仓库里删除而不会从文件系统里被删除掉。但是文件任然会在之前的版本之中，而之后的版本将不再存在。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help rm
Usage: fossil rm FILE...
   or: fossil del FILE...
Remove one or more files from the tree.
This command does not remove the files from disk.  It just marks the
files as no longer being part of the project.  In other words, future
changes to the named files will not be versioned.
```

Figure 5.4.: rm命令

你可以使用通配符删除一组文件，因此如果我有一组文件如com_tr.c, com_rx.c 和com_mgmt.c ，我可以使用如下命令删除它们：

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

通过运行“fossil status”命令你可以看到在下一次提交时将会被删除的文件。

5.2.4. rename or mv

这个命令用来将仓库里的一个文件重命名。这个命令不会讲磁盘上的文件重命名，所以一般用来在你已经将磁盘上的文件重命名，然后你想要在仓库里对其重命名时。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help rename
Usage: fossil mv|rename OLDNAME NEWNAME
   or: fossil mv|rename OLDNAME... DIR
Move or rename one or more files within the tree
This command does not rename the files on disk.  All this command does is
record the fact that filenames have changed so that appropriate notations
can be made at the next commit/checkin.
```

Figure 5.5.: rename命令

和add和rm命令一样你可以在名称里通配符来对一组文件进行重命名，同样的使用“fossil status”命令可以查看当前的状态。

5.2.5. status

命令status用来显示与仓库相关文件的状态。它将为你显示所有文件的增加、删除、修改的状态。这里有一个条件就是只会显示已经在仓库里的文件的状态或者说是已经在fossil控制下的文件。它还会显示你检出的版本在时间线的位置以及你仓库保存的位置。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/FossilBook.fossil
local-root:   /Users/jschimpf/Public/FossilBook/
server-code:  3e67da6d6212494456c69b1c5406a277d7e50430
checkout:     edd5b5fa4277604f365ec09238422c0aa7a28faf 2010-05-08 14:44:21 UTC
parent:       3f019cbc730db0eb35f20941533a22635856b2b3 2010-05-08 11:15:19 UTC
tags:         trunk
EDITED     fossilbook.lyx
EDITED     outline.txt
```

Figure 5.6.: status命令

上述的清单显示了我克隆的仓库代码副本的保存位置，我的工作位置，Tags标签显示我检出的是主分支trunk。最后显示文件的状态：我正在其中二个文件中进行工作。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help changes
Usage: fossil changes
Report on the edit status of all files in the current checkout.
See also the "status" and "extra" commands.
```

Figure 5.7.: changes命令

5.2.6. changes

这个命令列出所有修改的文件，和status命令类似，但是不会显示satus会显示的其它信息。

5.2.7. extra

命令extra用来查看你在你工作目录已经增加而没有在Fossil的控制之下的文件。这个很重要，因为如果你移动了你的工作目录或者其它人使用这个仓库时他们将无法获得这些文件。

```
Pandora-2:jschimpf/Public/FossilBook] jim% fossil help extra
Usage: fossil extras ?--dotfiles? ?--ignore GLOBPATTERN?
Print a list of all files in the source tree that are not part of
the current checkout.  See also the "clean" command.
Files and subdirectories whose names begin with "." are normally
ignored but can be included by adding the --dotfiles option.
```

Figure 5.8.: extra命令

选项–dotfiles会显示所有以“.”开始却不在Fossil管理之下的文件。当你想要这类文件也加入到你的仓库时这个选项非常重要。最后一个选项是“–ignore”让你忽略那些你不想放入仓库的确定文件。在我的例子里有一个名为fossilbook.lyx~的文件，它是LyX的备份文件，我不想放入仓库，因为它是临时文件。那么我可以输入命令：

```
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~
```

5.2.8. revert

命令revert用来将一个文件回滚到仓库对应的文件。当你编辑错误或者犯其它错误时这个命令非常有用。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% 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 a file is reverted accidently, it can be restored using
the "fossil undo" command.
```

Figure 5.9.: revert命令

如果不添加任何参数将回滚到当前的版本，见图解 5.6 位于 51页。选项-r允许你从时间线选择一个版本进行回滚。

5.2.9. update

命令update用来更新一个文件或所有文件到仓库的最新版本。在多人工作的模式下，你必须在每天开始工作前执行此操作，以保证你拥有所有文件的最新版本。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help update
Usage: fossil update ?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 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.  If FILES is omitted, all
files in the current checkout are subject to be updated.
The -n or --nochange 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.
```

Figure 5.10.: update命令

Update命令有很多选项，第一个你可以用来指定更新到某个特定的版本，如果不添加这个选项默认检取最新的版本。第二个参数可以让你对一个文件或者许多文件执行操作。你可以执行如下命令：

```
fossil update *.c
```

它将会更新所有以.c结束的文件。

如果有大量的变更，它还有一个“演练”的模式。如果你增加-n参数，它会显示什么将会修改被做而不会真正地做变更。当你不确定将会发生什么事情的时候做这个尝试非常有用。选项-v（可以与-n一起使用或者单独使用）会打印出每一个文件包括没有任何变动的文件。

5.2.10. checkout or co

这个命令和update类似。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help checkout
Usage: fossil checkout VERSION ?-f|--force? ?--keep?
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.
See also the "update" command.
```

Figure 5.11.: checkout或co命令

5.2.11. undo

这个命令用来撤销上一个命令如update, merge, 或revert操作。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help undo
Usage: fossil undo ?FILENAME...?
Undo the most recent update or merge or revert operation.  If FILENAME is
specified then restore the content of the named file(s) but otherwise
leave the update or merge or revert in effect.
A single level of undo/redo is supported.  The undo/redo stack
is cleared by the commit and checkout commands.
```

Figure 5.12.: undo命令

它可以指定单独一个文件或者指定的一组文件，否则如果不指定文件，它会对从上一次修改撤销所有文件的修改。

5.2.12. diff

命令diff会产生一个显示当前文件与仓库文件所有不同的文字清单。如果你不指定文件，它会显示所有当前目录下文件与仓库对应文件的不同之处（会略过未修改的文件）。如果使用–form和–to选项，你可以指定二个仓库里的不同版本进行对比。如果不加–to参数意味着与当前工作目录作对比。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help gdiff
Usage: fossil diff|gdiff ?options? ?FILE?
Show the difference between the current version of FILE (as it
exists on disk) and that same file as it was checked out.  Or
if the FILE argument is omitted, show the unsaved changed 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 "-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".
```

Figure 5.14.: gdiff命令

如果你定义了第三方的diff程序它会自动使用，除非你使用-i参数使用Fossil内建的diff功能。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help diff
Usage: fossil diff|gdiff ?options? ?FILE?
Show the difference between the current version of FILE (as it
exists on disk) and that same file as it was checked out.  Or
if the FILE argument is omitted, show the unsaved changed 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 "-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".
```

Figure 5.13.: diff命令

5.2.13. gdiff

这个命令和diff命令一样，但是（前提是你定义了gdiff的程序）在你的操作系统里会显示在一个图形界面里。查看UI界面里的设置选项，阅读如何设置图形diff程序。

5.2.14. ui

命令ui会启动一个本地的网站服务器。选项–port可以指定一个端口，默认使用的是8080。它会自动启动你操作系统的默认浏览器并自动跳转到你仓库的网站页面。如果你

在工作目录里运行这个命令它会为自动打开当前目录下的仓库的页面。如果在其它文件夹你可以在命令行指定一个仓库。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help ui
Usage: fossil server ?-P|--port TCPPORT? ?REPOSITORY?
   Or: fossil ui ?-P|--port TCPPORT? ?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.
In the "server" command, the REPOSITORY can be a directory (aka folder)
that contains one or more repositories with names ending in ".fossil".
In that case, the first element of the URL is used to select among the
various repositories.
```

Figure 5.15.: ui命令

5.2.15. server

这个是一个比ui强大的命令。它允许你只运行一个Fossil网站服务器却可以同时托管多个仓库。这样不同于指定特定的仓库，你需要指定一个目录，这个目录里有大量的仓库（全部都是以后缀名fossil结束），这样你就可以使用其中的任何一个仓库了。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help server
Usage: fossil server ?-P|--port TCPPORT? ?REPOSITORY?
   Or: fossil ui ?-P|--port TCPPORT? ?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.
In the "server" command, the REPOSITORY can be a directory (aka folder)
that contains one or more repositories with names ending in ".fossil".
In that case, the first element of the URL is used to select among the
various repositories.
```

Figure 5.16.: server命令

5.2.16. commit or ci

这个命令用来将你工作目录里的文件的修改内容提交到仓库里，且会为提交创建一个新版本并更新时间线。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% 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
one of the "-m" or "-M" options are used to specify a comment.
"-m" takes a single string for the commit message and "-M" requires
a filename from which to read the commit message. If neither "-m"
nor "-M" are specified then 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.
You will be prompted for your GPG pass phrase in order to sign the
new manifest unless the "--nosign" options is used.  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 cases the new check-in
to be placed in the named branch.  The --bgcolor option can be 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.
A check-in is not permitted to fork unless the --force or -f
option appears.  A check-in is not allowed against a closed check-in.
The --private option creates a private check-in that is never synced.
Children of private check-ins are automatically private.
Options:
   --comment|-m COMMENT-TEXT
   --branch NEW-BRANCH-NAME
   --bgcolor COLOR
   --nosign
   --force|-f
   --private
   --message-file|-M COMMENT-FILE
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 5.17.: commit命令

每次提交都添加一个有意义的备注信息（使用-comment或-m参数）是一个非常好主意。这样你就可以在时间线里读到相关的文字内容。

5.3. 维护命令

这些命令相对使用不是很频繁，因为在一般的操作里这些操作很少会用到。你在使用前查看本文档或者使用**fossil help**命令查看具体描述很有必要。一些命令如new和clone只在你开始一个仓库时会用到。其它如rebuild或者reconstruct只是用来修改或者升级仓库。

5.3.1. new

这个命令用来创建一个新的仓库

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help new
Usage: fossil new ?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.
Options:
   --admin-user|-A USERNAME
   --date-override DATETIME
```

Figure 5.18.: new命令

文件名可以用来指定仓库的名称。选项允许你指定管理人员的名称以及开始日期，如当你想要使用指定的而不是当前的登陆用户、想要使用与现在不同的时间的时候。

5.3.2. clone

克隆命令用来从主仓库创建一个你自己的本地版本。如果你在使用基于网络访问支持多用户的源仓库时(见章节 3.2.1 位于 23页)，那么这个命令将会将托管的主仓库复制到你的本地机器。同时它将会在你的副本和主仓库直接建立连接，所以对你对本地仓库的修改会传递到主仓库。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help clone
Usage: fossil clone ?OPTIONS? URL FILENAME
Make a clone of a repository specified by URL in the local
file named FILENAME.
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
```

Figure 5.19.: clone命令

和create一样你可以为这个clone的仓库指定一个管理员。主仓库的URL地址可以使用如下格式:

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

参数**user**和**password**是你指定仓库中的一个有效用户及其密码。在clone之前最好使用浏览器查看一下你的账户和密码。保证你可以访问这个仓库，例如这本书的仓库地址如下:

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

将其输入浏览器的地址栏应该可以访问到这本书的主页(见图示 3.5 位于 25页)。在你已经验证成功后，clone操作应该可以正常工作了。

不要忘记（如我一直做的那样）输入本地仓库的名称（见上述FILENAME）。

5.3.3. open

打开命令用来将仓库里的文件复制到工作目录里。这样以后你可以创建或者修改产品了。命令还将仓库与工作目录关联起来，这样提交会自动将提交到仓库。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help open
Usage: fossil open FILENAME ?VERSION? ?--keep?
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.
See also the "close" command.
```

Figure 5.20.: open命令

如果在多用户模式或者你的仓库有多个分支的情况下，指定一个特定的版本非常明智。当你这样运行命令以后Fossil会自动在你工作区域创建所有的文件和目录。额外的文件_FOSSIL_，manifiest，manifest.uuid也会自动被创建出来。

5.3.4. close

这个命令和open刚好相反，这个命令会中断当前工作目录和Fossil仓库之间的连接。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help close
Usage: fossil close ?-f|--force?
The opposite of "open".  Close the current database connection.
Require a -f or --force flag if there are unsaved changed in the
current check-out.
```

Figure 5.21.: close命令

当你需要放弃当前工作目录是，这个命令非常有用。如果工作目录与仓库之间有变动Fossil不会让你执行这个操作。使用–force参数可以强制切断他们之间的连接而不管是否有变更存在。

5.3.5. version

这个命令用来显示当前Fossil的版本

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help version
Usage: fossil version
Print the source code version number for the fossil executable.
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil version
This is fossil version [c56af61e5e] 2010-04-22 15:48:25 UTC
[Pandora-2:jschimpf/Public/FossilBook] jim%
```

Figure 5.22.: version命令

上述图示展示了如何还有帮助命令及使用了上述命令。当你对Fossil有疑问时，获取你使用的fossil版本是非常重要的。然后你可以使用你获得的版本号在Fossil的新闻组查找你使用版本的相关问题，这样他们就可以很快回复你如何修复这个问题或者是一个新的问题。

5.3.6. rebuild

如果你将你的Fossil的版本更新了，你需要为你所有的仓库运行此命令。这个命令将会自动将你的仓库更新到你所使用的新版本Fossul。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help rebuild
Usage: fossil rebuild ?REPOSITORY?
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.
```

Figure 5.23.: rebuild命令

5.3.7. all

这个事实上是一个修改器，在确定的命令之前使用将会对所有的仓库执行指定的命令。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help all
Usage: fossil all (list|ls|pull|push|rebuild|sync)
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 disconnection operation.
Available operations are:
   list       Display the location of all repositories
   ls         An alias for "list"
   pull       Run a "pull" operation on all repositories
   push       Run a "push" on all repositories
   rebuild    Rebuild on all repositories
   sync       Run a "sync" on all repositories
Repositories are automatically added to the set of known repositories
when one of the following commands against the repository: clone, info,
pull, push, or sync
```

Figure 5.24.: all命令

5.3.8. push

这个命令会把本地仓库的所有修改推送到主仓库或者远程仓库。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help push
Usage: fossil push ?URL? ?options?
Push changes in the local repository over into a remote repository.
Use the "-R REPO" or "--repository REPO" command-line options
to specify an alternative repository file.
If the URL is not specified, then the URL from the most recent
clone, push, pull, remote-url, or sync command is used.
The URL specified normally becomes the new "remote-url" used for
subsequent push, pull, and sync operations.  However, the "--once"
command-line option makes the URL a one-time-use URL that is not
saved.
See also: clone, pull, sync, remote-url
```

Figure 5.25.: push命令

5.3.9. pull

这个命令将会从远程仓库把所有的更改拉取到本地仓库。你然后可以使用**update**命令将所有更改应用到检出的文件。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help pull
Usage: fossil pull ?URL? ?options?
Pull changes from a remote repository into the local repository.
Use the "-R REPO" or "--repository REPO" command-line options
to specify an alternative repository file.
If the URL is not specified, then the URL from the most recent
clone, push, pull, remote-url, or sync command is used.
The URL specified normally becomes the new "remote-url" used for
subsequent push, pull, and sync operations.  However, the "--once"
command-line option makes the URL a one-time-use URL that is not
saved.
See also: clone, push, sync, remote-url
```

Figure 5.26.: pull命令

5.3.10. sync

这个命令用来同步远程副本和源仓库的副本，它同时执行push和pull操作。也可以通过指定远程仓库的URL地址来将本地仓库同步到多个远程仓库。如果你执行update命令时增加-n参数来进行演练，你执行fossil sync，起先不会执行同步操作，直到你使用fossil update -n来执行上述操作。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help sync
Usage: fossil sync ?URL? ?options?
Synchronize the local repository with a remote repository.  This is
the equivalent of running both "push" and "pull" at the same time.
Use the "-R REPO" or "--repository REPO" command-line options
to specify an alternative repository file.
If a user-id and password are required, specify them as follows:
    http://userid:password@www.domain.com:1234/path
If the URL is not specified, then the URL from the most recent successful
clone, push, pull, remote-url, or sync command is used.
The URL specified normally becomes the new "remote-url" used for
subsequent push, pull, and sync operations.  However, the "--once"
command-line option makes the URL a one-time-use URL that is not
saved.
See also:  clone, push, pull, remote-url
```

Figure 5.27.: sync命令

5.3.11. clean

这个命令用来删除所有在源码树中的所有“extra”文件。当你要清理源代码或者创建一个干净的结构时这个命令非常有用。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help clean
Usage: fossil clean ?--force? ?--dotfiles?
Delete all "extra" files in the source tree.  "Extra" files are
files that are not officially part of the checkout.  See also
the "extra" command. This operation cannot be undone.
You will be prompted before removing each file. If you are
sure you wish to remove all "extra" files you can specify the
optional --force flag and no prompts will be issued.
Files and subdirectories whose names begin with "." are
normally ignored.  They are included if the "--dotfiles" option
is used.
```

Figure 5.28.: clean命令

5.3.12. branch

当你想要创建或者列出仓库的分支时使用这个命令。之前我们讨论过分叉(见章节 3.4.3 位于 29页)；和分支是一样的概念，但是在用户的控制之下。这个会出现在当你有版本1.0，但是想在版本2.0里增加新功能，并保持版本1.0分支用于维护时。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help branch
Usage: fossil branch SUBCOMMAND ... ?-R|--repository FILE?
Run various subcommands on the branches of the open repository or
of the repository identified by the -R or --repository option.
   fossil branch new BRANCH-NAME BASIS ?-bgcolor COLOR?
       Create a new branch BRANCH-NAME off of check-in BASIS.
       You can optionally give the branch a default color.
   fossil branch list
       List all branches
```

Figure 5.29.: branch命令

5.3.13. merge

这个命令和branch相对应，由于将二个分支合并起来。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help merge
Usage: fossil merge [--cherrypick] [--backout] VERSION
The argument 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.
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:
  --detail                Show additional details of the merge
  --binary GLOBPATTERN    Treat files that match GLOBPATTERN as binary
                          and do not try to merge parallel changes.  This
                          option overrides the "binary-glob" setting.
```

Figure 5.30.: merge命令

5.3.14. tag

这个命令可以用来控制“Tags”，也就是你在时间线上每个等级条目的参数。当然你也可以使用add/delete/controll在UI界面的时间线里来操作tags，选择任何一个条目然后编辑。参见 **??** 位于 ??页。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help tag
Usage: fossil tag SUBCOMMAND ...
Run various subcommands to control tags and properties
    fossil tag add ?--raw? ?--propagate? 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
    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.
    fossil tag find ?--raw? TAGNAME
        List all check-ins that use TAGNAME
    fossil tag list ?--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
will assume that "decaf" is a tag/branch name.
```

Figure 5.31.: tag命令

5.3.15. settings

这个命令用来设置或者取消设置Fossil的属性参数。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help settings
COMMAND: settings
COMMAND: unset
fossil setting ?PROPERTY? ?VALUE? ?-global?
fossil unset PROPERTY ?-global?
The "setting" 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.
The "unset" command clears a property setting.
   auto-captcha     If enabled, the Login page will provide a button
                    which uses JavaScript to fill out the captcha for
                    the "anonymous" user. (Most bots cannot use JavaScript.)
   autosync         If enabled, automatically pull prior to commit
                    or update and automatically push after commit or
                    tag or branch creation.  If the the value is "pullonly"
                    then only pull operations occur automatically.
   binary-glob      The VALUE is a comma-separated list of GLOB patterns
                    that should be treated as binary files for merging
                    purposes.  Example:   *.xml
   clearsign        When enabled, fossil will attempt to sign all commits
                    with gpg.  When disabled (the default), commits will
                    be unsigned.
   diff-command     External command to run when performing a diff.
                    If undefined, the internal text diff will be used.
   dont-push        Prevent this repository from pushing from client to
                    server.  Useful when setting up a private branch.
   editor           Text editor command used for check-in comments.
   gdiff-command    External command to run when performing a graphical
                    diff. If undefined, text diff will be used.
   http-port        The TCP/IP port number to use by the "server"
                    and "ui" commands.  Default: 8080
   ignore-glob      The VALUE is a comma-separated list of GLOB patterns
                    specifying files that the "extra" command will ignore.
                    Example:  *.o,*.obj,*.exe
   localauth        If enabled, require that HTTP connections from
                    127.0.0.1 be authenticated by password.  If
                    false, all HTTP requests from localhost have
                    unrestricted access to the repository.
   mtime-changes    Use file modification times (mtimes) to detect when
                    files have been modified.  (Default "on".)
   pgp-command      Command used to clear-sign manifests at check-in.
                    The default is "gpg --clearsign -o ".
   proxy            URL of the HTTP proxy.  If undefined or "off" then
                    the "http_proxy" environment variable is consulted.
                    If the http_proxy environment variable is undefined
                    then a direct HTTP connection is used.
   web-browser      A shell command used to launch your preferred
                    web browser when given a URL as an argument.
                    Defaults to "start" on windows, "open" on Mac,
                    and "firefox" on Unix.
```

Figure 5.32.: settings命令

5.4. 杂项

这些命令看起来不适合任何分类但是却非常有用。

5.4.1. zip

这个命令所做的和基于用户界面的一样。如图示 2.13 位于 13页你可以下载一个指定版本文件的ZIP压缩文档。这个命令使得你可以从命令行执行该操作。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help zip
Usage: fossil zip VERSION OUTPUTFILE [--name DIRECTORYNAME]
Generate a ZIP archive for a specified version.  If the --name option is
used, it argument becomes the name of the top-level directory in the
resulting ZIP archive.  If --name is omitted, the top-level directory
named is derived from the project name, the check-in date and time, and
the artifact ID of the check-in.
```

Figure 5.33.: zip命令

5.4.2. user

这个命令用于修改用户信息。这个命令复制了你在用户界面可以做的一些操作，详细可查看图示 3.6 位于 26页。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% 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
       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 5.34.: user命令

5.4.3. finfo

这个命令可以输出任意指定文件的历史信息。如果你在其它系统里需要使用这个文件的历史信息时这个命令非常有用。你可以将结果传递到其它可以解析和使用该数据的系统。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help finfo
Usage: fossil finfo FILENAME
Print the change history for a single file.
The "--limit N" and "--offset P" options limits the output to the first
N changes after skipping P changes.
```

Figure 5.35.: finfo detail

一个查勘我们这本书的仓库里的outline.txt文件历史的例子如下：

```
[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

这个可以多样地输出项目的时间线。如果你在为Fossil创建GUI界面及想要显示时间线时这个命令非常有用。你可以执行这个命令然后获取返回结果并在你的UI上显示。这里列出了很多选项来T控制显示的样式。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help timeline
Usage: fossil timeline ?WHEN? ?BASELINE|DATETIME? ?-n N? ?-t TYPE?
Print a summary of activity going backwards in date and time
specified or from the current date and time if no arguments
are given.  Show as many as N (default 20) check-ins.  The
WHEN argument can be any unique abbreviation of one of these
keywords:
    before
    after
    descendants | children
    ancestors | parents
The BASELINE can be any unique prefix of 4 characters or more.
The DATETIME should be in the ISO8601 format.  For
examples: "2007-08-18 07:21:21".  You can also say "current"
for the current version or "now" for the current time.
The optional TYPE argument may any types supported by the /timeline
page. For example:
    w  = wiki commits only
    ci = file commits only
    t  = tickets only
```

Figure 5.36.: timeline命令

5.4.5. wiki

这个命令允许你从命令行来控制Wiki。这个在你使用命令行控制Fossil或者想要增加大量的自动生成的页面到Wiki时会非常有用。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help wiki
Usage: fossil wiki (export|create|commit|list) WikiName
Run various subcommands to work with wiki entries.
    fossil wiki export PAGENAME ?FILE?
       Sends the latest version of the PAGENAME wiki
       entry to the given file or standard output.
    fossil wiki commit PAGENAME ?FILE?
       Commit changes to a wiki page from FILE or from standard
       input.
    fossil wiki create PAGENAME ?FILE?
       Create a new wiki page with initial content taken from
       FILE or from standard input.
    fossil wiki list
       Lists all wiki entries, one per line, ordered
       case-insentively by name.
TODOs:
    fossil wiki export ?-u ARTIFACT? WikiName ?FILE?
       Outputs the selected version of WikiName.
    fossil wiki delete ?-m MESSAGE? WikiName
       The same as deleting a file entry, but i don't know if fossil
       supports a commit message for Wiki entries.
    fossil wiki ?-u? ?-d? ?-s=[|]? list
       Lists the artifact ID and/or Date of last change along with
       each entry name, delimited by the -s char.
    fossil wiki diff ?ARTIFACT? ?-f infile[=stdin]? EntryName
       Diffs the local copy of a page with a given version (defaulting
       to the head version).
```

Figure 5.37.: wiki命令

5.5. 高级命令

这些命令你很少会用到。这些命令只有在使用Fossil做非常复杂的事情时才会被用到。如果你不得不使用这些命令，你大概已经超出了本书所讲述的范围。

5.5.1. scrub

这个命令用来从仓库删除敏感信息如密码。这样就运行你将整个仓库发送给某个人而不必担心你的个人信息。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help scrub
COMMAND: scrub
fossil scrub [--verily] [--force] [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.
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.
```

Figure 5.38.: scrub命令

5.5.2. search

这个命令为一个样品搜索时间线。这个也可以在界面的时间线页面来操作。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help search
COMMAND: search
fossil search pattern...
Search for timeline entries matching the pattern.
```

Figure 5.39.: search命令

5.5.3. sha1sum

这个命令可以指定文件计算SHA1的值。这个结果Fossil使用了所有对象的进行计算，可以保证每个任何文件都有一个唯一值。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help sha1sum
COMMAND: sha1sum
fossil sha1sum FILE...
Compute an SHA1 checksum of all files named on the command-line.
If an file is named "-" then take its content from standard input.
```

Figure 5.40.: sha1sum命令

5.5.4. rstats

为当前检出的仓库打印一份状态报告

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help rstats
Usage: fossil rstats
Deliver a report of the repository statistics for the
current checkout.
```

Figure 5.41.: rstats命令

比如，在Fossil用户向导仓库里运行它：

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil rstats
 Number of Artifacts: 137
  59 full text + 78 delta blobs
 278961 bytes average, 38217738 bytes total
  Number Of Checkins: 26
     Number Of Files: 37
Number Of Wiki Pages: 2
   Number Of Tickets: 6
 Duration Of Project: 23 days
```

5.5.5. configuration

这个命令运行你保存或者加载Fossil的定制配置

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help configuration
Usage: fossil configure METHOD ...
Where METHOD is one of: export import merge pull push reset.  All methods
accept the -R or --repository option to specific a repository.
   fossil configuration export AREA FILENAME
        Write to FILENAME exported configuration information for AREA.
        AREA can be one of:  all ticket skin project
   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.
   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.
   fossil configuration reset AREA
        Restore the configuration to the default.  AREA as above.
WARNING: Do not import, merge, or pull configurations from an untrusted
source.  The inbound configuration is not checked for safety and can
introduce security vulnerabilities.
```

Figure 5.42.: configuration命令

5.5.6. descendants

使用这个命令来查看文件在时间线的什么位置。

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil help descendants
Usage: fossil descendants ?BASELINE-ID?
Find all leaf descendants of the baseline specified or if the argument
is omitted, of the baseline currently checked out.
```

Figure 5.43.: descendants命令

6. Fossil定制 - TH脚本语言

6.1. TH简介

6.1.1. TH是一个类Tcl语言

TH是一个基于TCL语言和字符的命令式语言。这个语言只有很少的基本构造和相当简单的语法，也就是非常简单的意思。TH被设计为“Fossil”源码管理系统的脚本语言。TH是一个解释型语言，它在执行脚本时进行解析、编译和执行。在Fossil里，TH脚本的典型应用时创建内建网站的页面，经常用于对表单提交进行响应。

TH的基础机制都与字符和字符替换有关联。TH/Tcl的做事情的方式于其他一些你已经很熟悉的语言相比有一些不同，所以保证你理解基本概念是非常有价值的。

6.2. “Hello, world”程序

按传统，开始介绍一个语言的一个标准用例是"Hello, World"程序。在TH里这个程序只有一行：

```
puts “Hello, world\n”
```

在TH里输出一个字符的命令是“puts”，Puts命令后面的单独的文本内容会被打印到输出流中。如果字符多余一个单词，你必须使用双引号或者波浪号将其包裹在内。引号或者波浪号内的单词会被视为一个单独的单元，当文字被空格拆分后，会被当做命令的其它参数使用。

6.3. TH结构和语法

6.3.1. 数据类型

TH内部核心中只有一个数据类型，也就是字符串。TH里的所有值都是字符串，变量保留字符串并生成返回字符串。 TH里的字符由单字节的字符组成以零结束。超过ASCII区间的字符，例如在区间0x80-0xff内的字符对于TH来说没有意义：它们不考虑数字，字符甚至是空格。

基于上文所述，TH可以有四种方式解释一个字符串。第一是就是由一连串的任意字符组成；第二是将字符认为是一个列表，一个有顺序的由空格分隔的单词组成；第三是一个

字符可以是一个命令,一个命令的意思是第一个单词被认为是一个命令，后面跟随的是一连串的参数；第四，最后一种，字符可以被解释为一个表达式。

下面我们详细讨论字符串前三个解释方式。

6.3.2. 列表

TH里的一个列表由一串排列顺序的对象或者被空格分隔的单词组成。TH中下面的字符由空格组成。

```
logo FossilBook Artifact Content Logged in as frans Home
Timeline Files Branches Tags Tickets Wiki Admin Logout
Download Hex Shun Artifact
7b03c1c83fcd092b090ada102ed76356e9496f2d
File fossilbook.lyx 2010-06-26 12:28:12 - part of checkin
[9ad19c40d6] on branch trunk - Added section on TH
scripting [b832f46d31] (user:  jim) [annotate]
logo FossilBook Artifact Content Logged in as frans Home
Timeline Files Branches Tags Tickets Wiki Admin Logout
Download Hex Shun Artifact
7b03c1c83fcd092b090ada102ed76356e9496f2d
File fossilbook.lyx 2010-06-26 12:28:12 - part of checkin
[9ad19c40d6] on branch trunk - Added section on TH
scripting [b832f46d31] (user:  jim) [annotate]
     ' '    0x20
     '\t'   0x09
     '\n'   0x0A
     '\v'   0x0B
     '\f'   0x0C
     '\r'   0x0D
```

一个单词会是任何由空格限定的字符序列。单词不管是字母还是数字的：“.%*”在TH里是一个有效的单词。简单地说一个单词包含嵌入的空格字符，它需要被引号标识，不管是双引号或者开始结束的波浪号。引号有将引号内的内容放入单独的列表元素里的效果。

单词不能以TH的特殊字符{ } [ ] \ ; 和 "开始。注意单引号‘在TH里不是一个特殊字符。必须避免使用这些字符作为字符串开始，后面将深入进行讨论。

TH提供了很多内建的命令用于列表操作，如计算单词的字符数量，使用列表索引检索或添加单独的单词。这些命令会在章节“使用列表”中进行讨论。

6.3.3. 命令

TH将所有的东西都按命令的模式来解释，甚至是程序结构如变量分配和过程定义。TH只增加了非常少的语法来正确地调用命令，然后就可以将工作重点移动到命令的实现上来。

命令是一个特殊格式的列表。基本的TH命令语法如下：

```
command arg1 arg2 arg3 ...
```

这个命令不管是内建的命令或者是一个TH过程。

空格用来将命令名称和它的参数区分开来，一个新行或者分号可以用来结束一个命令。TH的注释是以符号“#”开始的行，或者是在一行命令结束的分号后面添加符号“#”然后进行注释。

6.3.4. 分割 & 转义

TH不会解释命令的参数来执行分割，这里允许在一个参数里有多个单词，转义用于处理特殊字符以提取变量来执行内嵌的命令。因此，TH命令处理程序的表现可以概述为下述三个步骤。

1. 参数分割
2. 值替代为反斜杠、变量和内嵌命令
3. 命令调用

注意没有评估命令参数的步骤。完成替换后，参数会逐字地传递给命令而由命令来验证它的参数是否需要。

6.3.4.1. 参数分割

TH有二个机制将多个单词分割成单独的对象：

- 双引号, “ ”
- 花括号, { }

它们都有将一组单词分组的效果，它们在下一阶段的转义时有不同的影响。大概是，双引号将内容分组而花括号会分组且会将阻止内容里的转义。

分组从字符的左边开始到右边且不会影响后面转义。如果一个转义指向了一个可以不同分割的字符时，它不会起作用，因为其在分割阶段已经决定了。

6.3.4.2. 值转义

TH执行三个不同的替换(查看th.c/thSubstWord代码获取详细信息)

- 转义反斜杠
- 转义变量
- 转义内嵌的命令

和分割一样，转义也是从左边向右边且只执行一次：如果一个转义指向了一个还可以被转义的字符串时，不会有任何效果。

6.3.4.3. 反斜杠转义

总体来说，反斜杠(\)会禁止替换反斜杠后面的字符串。任何反斜杠后面的字符任然会保持原样。 当想要忽略特殊意思的字符{ } [ ] \; 和"时反斜杠非常有用。

有二个在转义阶段会被转义成特殊字符的字符。一个反斜杠跟着一个字母'n'会被转义为新行符号，和C语言一样。一个反斜杠跟随一个字母'x'及二个十六进制数字会被转义为其对于的字符。例如：写"\x20"和输入一个空格是一个意思。需要注意字母\x转义不会"一直"转义反斜杠后面的值，除非他还是Tcl里的十六进制值，需要强调的是只有二个值。字符串\x2121不是一个单独的感叹号，而是三个字符!21。

6.3.4.4. 变量转义

和其它编程语言一样，TH也有变量的概念。TH变量是一个保存字符串值的命名容器。变量会在下面的文章里详细探讨，我们将目前讨论的范围限制在变量转义。

美元符号($)可能用来作为转义变量值的特殊写法。如果$出现在一个参数里且没有用花括号关闭那么会执行变量转义。美元符号$后面的字符串如上文所述不是一个数字、字符或者下划线。会被认为是一个变量名称，那个变量所对应的字符串值会被根据其名称进行转义。比如：如果变量foo的值为test，那么命令"puts $foo.c"等同于下文的命令：

```
"puts test.c"
```

变量转义有二种特殊的格式。如果变量的名称的下一个字符是一个开启的圆括号，那么这个变量会被认为是一个数组，圆括号内的字符都会认为是一个数组的索引。命令转义和变量转义会在将圆括号内的信息作为索引来执行。例如：如果变量x是一个数组且有一个名为index且值为87和另外一个名为14值为more的两个元素，那么命令：

```
puts xyz$x(first)zyx
```

等同于命令：

```
puts xyz87zyx
```

如果变量index有一个值"14"，那么命令：

```
puts xyz$x($index)zyx
```

等同于命令：

```
puts xyzmorezyx
```

查看上文变量章节和下文的数组章节以了解数组的更多信息。

第二个变量的实现格式是当美元符号$后面跟着花括号，这时变量名称为括号开始到结束的所有字符串。这个格式中不能使用数组引用：花括号内的指向的是一个标量。例如：如果变量foo有一个值叫"test"，那么命令：

```
set a abc${foo}bar
```

等同于命令：

```
set a abctestbar
```

一个美元符号后面加上其他的如字母、数组、下划线或者左括号会当作文字美元符号。下面的命令输出一个单独的$。

```
puts x $
```

6.3.4.5. 命令转义

最后一个构造是命令转义。一个内嵌的命令用方括号[ ]来限定其内容。TH解释器会读取括号内的所有内容并将其作为命令来处理。它会复述括号外的内容并将括号及其内的内容替换为内嵌命令的运行结果。

例子：

```
puts [string length foobar]
=> 6
```

在这个例子里，内嵌的命令式是：string length foobar 。这个命令返回字符foobar的长度值。内嵌命令首先运行，然后，命令转义触发外部命令并替代为其计算结果：

```
puts 6
```

如果在一个命令里有多个命令转义触发点，解释流程的顺序是从左至右。每当遇到一个右括号就认为该命令就此结束。结果会明确地排序，即哪个内嵌命令先执行然后将其结果作为外部命令的参数。

6.3.4.6. 再述参数分隔

在命令评估转义阶段，波浪号和双引号二组运算符在TH解释器里的处理不相同。双引号仅用于当内容在空格后的分割，字符a”b”是一个含有四个字符的字符串，而不是二个字符ab。

```
puts a″b″
=> a″b″
```

分割在波浪号内的单词时，会禁止括号内的转义。再次提示，一个开放的花括号仅在其在空格后才会进行分隔。花括号内的字符串会直接传递给命令，甚至是在反斜杠处理完成后。

需要注意的是花括号的这个效果只有其被作为分割时才有效（如：在单词序列的开始和结束）。如果一个字符串已经分割，不管是双引号还是花括号，或花括号在一个分割的字符串中间（如："foo{bar"），花括号会被作为普通字符处理而没有任何特殊意义。如果字符串使用双引号进行分割，引号内的字符会触发转义，甚至是在括号之间。

方括号语法用于命令转义而不含分割功能。作为替代，内嵌的命令会被认为是当前分割字符的一部分。在下面的命令，双引号将最后的参数分割了，但内嵌命令只是那个分组的一部分：

```
puts "The length of $s is [string length $s]."
```

如果一个参数仅有一个内嵌命令组成，那么你不需要使用双引号来分割，因为TH语法会将整个内嵌命令一个分组的一部分进行处理。一个内嵌命令会作为一个完整有序的字符处理，而不管其内部结构。在采集主命令的参数时，它包含了分组字符周边的字符。

6.3.5. 概要

下面的规则概述了TH解释器在调用命令之前进行分割和转义的基本机制：

- 命令参数使用空格进行分隔，除非参数使用花括号或者双引号使用如下方式进行分割。
- 使用花括号进行分割时，{}会阻止转义。花括号嵌套。解释器匹配分组内从左至右所有的字符，包括新行、分号、嵌套花括号。编码（如：最外部的）花括号不包含在组内。
- 使用双引号分割时，""，允许转义。解释器会将所有的东西进行分割，直到发现另外一个双引号，包含新行、引号。嵌入的引号不包含在字符的分组内。可以使用反斜杠使双引号字符放进一个分组（如：\"）。
- 分割操作会在转义之前执行，也就是说变量的值或者命令的执行结果不会影响分割。
- 美元符号$会触发变量转义。变量名可以由任意长度的字母组成。如果变量引用了其它嵌套的字符串，或者如果其包含其它如字母，下划线，可以使用${varname}语法进行区别。
- 方括号[]会触发命令转义。在括号内的所有东西都会被作为命令进行处理，命令里的所有内容都会被命令执行结果替换掉。允许嵌套。
- 反斜杠\用来引用特殊字符。你可以认为它是另外一种格式的转义，反斜杠及其后面的字符会被一个新的字符替代。
- 转义会在任意地方触发，除非使用花括号阻止。分组的一部分可以是常量字符串，另外的部分可以是转义后的结果。甚至是命令的名称也可以使用转义处理。
- 一个单独的转义会在命令引用之前执行。转义的结果不会被再次解释。如果你有包含特殊字符如空格、美元符号、方括号或者花括号的变量值或者命令结果时这个规则会很重要，因为只会执行一个循环的转义，而你不用担心特殊字符的值会触发一个额外的转义。

6.3.5.1. 警告

- 一个比较普遍的错误是在使用花括号或引号对参数分割时忘记参数间的空格。这是因为当使用花括号或引号来进行分割时，空格被当做了间隔符。如果你忘记空格，会引起一个关于参数数量错误的语法错误。下面的错误是因为在 } 和 { 之间忘记了空格：

```
if {$x > 1}{puts "x = $x"}
```

- 当使用双引号进行分割时，花括号的特殊效果会被关闭。双引号格式内的任意地方都会触发转义。在下面的命令里，变量仍然会被转义：

```
set x xvalue
set y "foo {$x} bar"
=> foo {xvalue} bar
```

- 使用方括号进行命令转义时不需要使用空格。在进行分割时，解释器会将方括号内的所有东西识别为当前分组的一部分。

6.4. TH表达式

TH解释器本身不会执行数学表达式。TH只做分割、转义和命令调用。然而，几个内建的命令见到一个或多个参数作为表达式时，会请求解释器计算此类表达式的值。

命令**expr command**是一个最简单的此类命令，用于解析并执行表达式：

```
puts [expr 7.4/2]
=> 3.7
```

注意表达式可以包含空格，但是如果有的话必须进行分割以将其识别为一个单独的参数。

TH在执行表达式时表达式的内容可使用三种数据类型：二种类型的数字，布尔值和浮点数，字符。正整数会转化成浮点值。布尔值True和False也可以使用正整数值1和0来替代。expr的实现非常小心地处理精确度并避免不必要的字符与数字的转化。

在表达式起效之前，变量和命令转义都会在表达式字符上执行。因此，你可以在数学表达式内包含变量引用及内嵌命令，甚至是在表达式字符包含在花括号内时。注意反斜杠转义在表达式里不会被执行。

TH表达式由操作数、运算符及圆括号组成。在它们之间可能会使用空格；空格会被表达式处理器忽略。当可能时，操作会被解读成整数值。如果一个操作数不是正整数格式，如果可能的话它会被解读成浮点数。浮点数在ANSI-标准的C编译器里会被通过多个方式规定。比如，下面的都是有效的浮点数：2.1，3.，6e4，7.91e+16。如果不能进行数值化转换，那么操作数会被认为是字符（只有少数的运算符可以使用它）。

操作数会应用下面任意一种方式被解读：

- 数字值，不管整数还是浮点数。
- 花括号内的字符串，括号内的字符串会被作为操作数使用且不会被转义。
- 双引号内字符串，表达式剖析引号内的信息的变量和命令转义，使用它们的结果作为操作数来执行。
- TH变量，使用$声明，变量的值会被作为操作数来使用。
- 方括号里的TH命令，命令会执行且它的结果会被作为操作数来使用。

如上触发转义的地方（例如：引号内的字符），它们会被处理器执行。然而，在表达式处理调用前，还有额外的一层转义可以会被命令剖析器执行。如下讨论，一般最好使用花括号来分隔表达式以阻止命令剖析在内容里执行转义。

有效的运算符如下，根据优先级由高到低排列：

运算器	动作
+ - ~ !	一元加、一元减、比特位NOT、逻辑NOT。这些运行都不会运用在字符操作数上，比特位NOT可能会被应用于整数。
* / %	乘法、除法、余数。它们都不会被引用在字符操作数上，余数只会被应用于整数。
+ -	加和减。只用于数值操作数验证。
<< >>	左移和右移。仅用于整数操作数。
< > <= >=	布尔运算小于、大于、小于等于和大于等于。每个运算符都会在条件为真时返回1为假时返回0。这些运算符会应用于字符操作数，同样也可以用于数值操作数，如字符串对比的案例。
== !=	布尔运算等于和不等于。同上该运算返回0/1结果。可用于验证所有的操作数类型。
eq ne	比较二个字符串是相同(eq)还是不同(ne)。它返回1 (true)或者0 (false)。这个运算可以用于验证字符串，但是不包含数字:
&	比特位AND，只用于整数操作数。
^	Bit-wise XOR. Valid for integer operands only
|	比特位OR，只用于整数操作数。
&&	逻辑计算AND，产生一个返回结果，1表示二个操作数都不是零，否则为0。仅用于整数操作数。注意：没有“求值捷径”： 尽管左侧求得值为False还是会求右侧的值。
||	逻辑计算OR，产生一个返回结果，1表示二个操作数都不是零，否则返回0。仅用于整数操作数。注意：没有“求值捷径”： 尽管左侧求得值为False还是会求右侧的值。

所有优先级相同的二进制运算都从左至右执行。比如表达式“ 4*2 < 7 ”返回值为0。

所有牵涉到整数的内部计算都使用的C语言的int类型，所有牵涉浮点的内部运算都是使用的C语言的double类型。所有的内部整数、浮点、字符操作数的转换都会根据需要自动执行。算法运算时，直到浮点被引入且点被使用之后才会使用整数。

字符串值可能作为比较处理器的操作数使用，尽管表达式会尝试进行整数和浮点的转换。如果相比较的操作数一个是字符另外一个是数值，那么数值操作数会被转回字符。

6.5. TH变量

和所有的编程语言一样TH也有变量的概念。TH变量绑定一个变量名到一个字符串值。变量名在其范围内必须唯一，不管是全局或者本地范围。TH支持二种类型的变量：常

量和数组。

TH允许通过使用$样式、set命令或其它少数构造的变量转义来定义和使用变量。变量不需要预定义：当一个新的变量名被使用时会自动创建一个变量。

6.5.1. 使用变量

TH有二个关键命令用于操作变量，set和unset：

```
set varname ?value?
unset varname
info exists varname
```

set command命令返回变量名对应的值。如果变量不存在，那么会抛出一个错误。如果可选参数并指定，那么set命令设置*varname*的值。如果不存在，那么在当前范围内创建一个新的变量并返回它的值。

unset command命令会从其范围内删除一个变量。参数*varname*是变量的名称。如果*varname*引用了一个数组的元素，那么那个元素会移除而不会影响剩余的数组元素。如果*varname*有一个数组的名称组成且不含括号的索引值，那么整个数组会被删除。**unset command**会返回一个空字符作为返回值。如果变量不存在那么会引起一个错误。

命令**info exists command**当变量名*varname*存在于当前范围是返回“1”，不管是在全局还是当前范围；否则返回“0”。

6.5.1.1. 标量变量和数组变量

TH支持二种类型的变量：标量和数组。一个标量变量只有一个单独的值， 反之一个数组变量可以有任意数量的元素，每一个都有对应的名称（命名为它的"索引"）及值。TH数组是一个可标识关联的数组，即索引可以是任意的字符串。

如果*varname*包含一个左圆括号并以右圆括号结束，那么它索引了一个数组元素：括号前的字符是数组的名称，括号内的字符是它的索引。否则*varname*就定义为一个标量变量。

比如，命令 **set [x(first) 44]**会将变量X索引为first的元素的值设置为新值44。TH中的二个标识数组可以使用指数模拟多个串联值。比如如下命令：

```
set a(2,3) 1
set a(3,6) 2
```

设置指数为2,3和3,6的元素。

大概地说，TH中数组元素可以在任意标量也被使用的地方使用。如果一个数组并定义为一个特殊的名称，那么就不可以使用该名称同时（同范围）定义一个标量变量。同样的，如果有一个标量变量使用了特定名称，那么就不能使用其作为数组变量使用。要转化一个标量为数组，应使用**unset**命令删除存在的变量，反之亦然。

6.5.1.2. 变量范围

变量存在于一个范围内。TH解释器维护了一个全局范围，就是可以为所有的命令执行时可以共享的变量。每一个用户自定义命令的调用都会创建一个本地范围。这个本地范围保存参数及本地变量，仅存在于用户命令的执行期间。

如果不在用户命令的实现主体，那么对varname的引用会指向一个全局的变量，如，一个在全局范围的变量。相反的，在用户自定义命令的实现主体里对varname的引用指向了一个参数或者命令的本地变量。然但是，在用户自定义命令的实现主体部分可以使用在全局变量前使用"::"来明确指定。

TH提供了一个特殊命令来连接不在当前命令的本地范围而是命令调用链的本地范围来限制到当前命令的范围里。这个命令是upvar:

```
upvar ?frame? othervar myvar ?othervar myvar ...?
```

命令**upvar command**将当前程序的一个活多个本地变量排序指向一个封闭的程序调用，或者指向全局变量。如果*frame*是正整数，那么就给出一个移动的距离（提升命令调用堆栈）。如果*othervar*不是正整数，参数*frame*可能会省略（之后frame默认是"1"）。对于每一个*othervar*参数，**upwar**命令在本地范围通过名称可辨别、*myvar*对应参数的给定的名称以及在当前程序通过*frame*偏移可连接性来构造变量。由*othervar*命名的变量不需要在调用时就存在；它会在myvar的第一次引用时被创建，就像普通变量一样。*upvar*命令只在用户定义命令的主体部分有意义。不管是Othervar还是myvar都不会指向数组的元素。*upvar*命令用以简化*call-by-name*程序调用的实现也使得创建一个新TH命令的控制结构变得简单些。比如，考虑如下程序:

```
proc incr {name} {
    upvar $name x
    set x [expr $x+1]
}
```

incr通过一个给定名称的变量进行了调用，它增加了那个变量的值。

6.6. TH命令，脚本和程序流程

在TH里事实上在命令（通常在其它语言里理解为'statements'或"functions"）和"syntax"没有差别。没有类似C、Java、Python、Perl之类语言的保留词汇。当TH解释器开始发现有一系列内建的、已知的命令那么解释器一般会解析一行代码。这些命令包含for、set、puts等等。他们包括所有的TH命令，然而，TH命令依然只经常遵从同样语法规则，无论是内建或那些你使用proc命令创建的TH命令。

6.6.1. 再述命令

和Tcl一样，TH由命令构成。一个命令为你做某些事情，例如输出一个字符，计算一个数学表达式，或者生成HTML以在屏幕显示一个部件。

命令是列表的的一个特殊形式。TH命令的基本语法如下:

```
command arg1 arg2 arg3 ...
```

command不管是内建命令还是一个TH程序。

使用空格来分隔命令及其参数，用新行符号或者分号来结束一个命令。TH语言的行注释使用符号“#”及后面的字符串构成，或者在命令结束分号后面添加一个“#”及注释文字。

6.6.2. 脚本

一般情况下，TH里的控制遵循从一个命令到下一个命令的顺序。下一个命令也可以在同一个列表里（若当前命令使用分号结束时）或在下一行。因此TH程序是一个有TH命令组成的列表。

这样的命令列表指向了一个“script”。一个脚本因此是一个自包含的代码片段由一个或多个命令构成。 脚本里的命令类型其它语言里的声明。

有些命令使用一个活多个脚本作为参数，取决于其它的参数运行这些脚本零或多次。比如，如果命令执行那个脚本或者其它脚本一次，取决于表达式的执行结果为True或False。命令执行脚本会执行一般的分割及转移，作为执行脚本的一部分。

> 请注意脚本都需要使用花括号来结束以阻止转义执行第二次：一次作为执行上层命令的一部分及在准备脚本时执行一次。忘记使用花括号关闭脚本参数是常见的源码错误。

少数的命令(return, error, break and continue)会立即停止当前执行的脚本，并将控制权交给列表的下一个命令。控制取代了返回到了当前脚本的初始化了的命令执行。

6.6.3. 命令结果代码

每个命令都会产生二个结果：一个结果代码和一个字符串。代码显示不管是命令成功或者不成功，字符串会给出额外的信息。有效的代码包含在“th.h”头文件里，如下：

名称	值	含义
TH_OK	0	这个是普通的返回代码。字符串会给出命令的返回值
TH_ERROR	1	表明发生了一个错误；字符包含了错误信息的描述
TH_RETURN	3	表明调用了返回命令，字符包含程序或命令的返回值
TH_BREAK	2	表明内部的循环将立即结束，字符包含了 “break”或者中断的参数，如果有的话
TH_CONTINUE	4	表明内部循环将进入下一次迭代。字符包含了“break”或者中断的参数，如果有的话

TH程序员一般不会考虑返回的代码，因为总是会返回TH_OK结果。如果其它结果被返回，那么TH解释器会立即停止命令执行并返回结果给调用的代码。如果TH解释器里有多个内嵌的调用进程，那么这些进程都会返回错误结果给调用代码，直到最终的报告被提交到程序代码的顶层。然后程序会为用户显示错误信息。

在少数情况下，一些命令会自己处理已知的 "错误"条件且不会向上层返回结果。比如，for命令会检查TH_BEREAK代码；如果触发了，for会停止执行主体循环并返回TH_OK给调用代码。for命令还会处理TH_CONTINUE 代码并交由解释器处理TH_RETURN代码。catch命令允许TH程序捕捉错误并处理他们且将会不会警告命令。

6.6.4. 流程控制命令

TH里的流程控制命令如下：

```
if expr1 body1 ?elseif expr2 body2? ? ?else? bodyN?
for init condition incr script
break    ?value?
continue ?value?
error    ?value?
catch script ?varname?
```

下面按顺序讨论每一个命令

if 命令的语法如下：

```
if expr1 body1 ?elseif expr2 body2? ? ?else? bodyN?
```

参数**expr**是表达式，body参数是脚本，elsif和else是参数是关键字常量字符。if命令选择body脚本执行来执行。

expr参数必须求解为一个正整数值。如果求值为非零值，执行后面body脚本且从根据脚本执行的返回继续if命令后面的命令。如果expr参数求值为零，它的body脚本会被跳过且尝试下一个选项。当没有更多的选项可以尝试时，任然继续执行下一个命令。

if命令返回执行脚本的值，当没有执行脚本时返回“0”。

for 命令的语法如下：

```
for init condition incr body
```

init，incr 和body参数都是脚本。条件参数是一个产生一个正整数结果的表达式。for命令是一个循环命令，和C语言里的声明结构类似。

for命令在TH解释器被调用执行init命令。然后它重复地将表达式作为条件进行求解；如果结果是非零它在TH解释器里调用body，然后再调用incr，再然后重复循环。当测试求值为零时命令终止。

如果一个continue命令在body脚本中被调用那么任何当前执行body里的剩余命令将会被跳过；流程会继续执行incr，然后求解条件，在继续。如果一个break命令在body或者next里被调用，那么for命令将会立即返回。break和continue的执行方式和C语言中类似声明的执行方式非常相似。

for命令返回一个空白字符。

break 命令的语法如下：

```
break ?value?
```

break从当前程序（或顶层的命令）命令立即返回，将结果作为返回值和TH_BREAK作为返回代码返回。如果结果没有特殊指定，字符“break”将作为结果返回。

continue 命令的语法如下：

```
continue ?value?
```

continue命令从当前程序（或顶层命令）立即返回，将值作为返回值和TH_CONTINUE作为返回代码返回。如果值没有特别指定，那么字符“continue”将会被作为结果返回。

error 命令的语法如下：

```
error ?value?
```

error命令从当前程序（或顶层命令）立即返回，将值作为返回值、将TH_ERROR作为结果返回，如果值没有特别指定，那么字符“error”将会被作为结果返回。

catch 命令的语法如下：

```
catch script ?varname?
```

catch命令可能用来阻止命令解析中止错误。catch命令递归地调用TH解释器执行脚本，且永远都返回TH_OK代码，不管执行脚本时发生何种错误。

catch的返回值是一个十进制的字符包含TH解释器执行脚本后返回的代码，如果命令没有错误它可能是”0“（TH_OK），否则它是上文表格所对应结果代码的一个非零值。如果给定varname参数，那么它将给出变量的名称；catch从运行的脚本设置返回变量的值（或者一个结果或一个错误信息）。

6.6.5. 创建用户自定义命令

proc 命令可以创建一个新命令。proc命令的语法如下：

```
proc name args body
```

proc命令创建一个新的TH命令程序，名称，使用名称取代任意存在相同名称的命令。不管何时调用命令，TH解释器将会执行命令包体里的内容。

参数args指定了程序正式参数。它考虑列表，可能为空，谁的每一个元素指定一个参数。每个参数指定也是一个列表无论一个还是二个单元。如果只指定一个单，那么它就是参数的名称；如果有二个单元，那么第一个是参数的名称第二个是它的默认值。花括号和反斜杠可能会被用来特别指定完整的默认值。

proc命令返回一个null字符串。

6.6.6. 执行用户自定义命令

如果一个命令是用户自定义的命令（如使用proc命令创建的命令），那么TH解释器会创建一个本地变量环境，绑定合法参数到其真实的值上（如TH唯一地使用调用值）然后加载脚本的包体。执行然后处理脚本的第一个命令。当最后一个命令结束或者返回命令被执行后脚本执行结束。当脚本结束后，本地变量环境会被删除；用户自定义命令后面的命令将会继续被执行。

更详细地说，当调用一个用户自定义命令时，本地变量环境为程序的每一个合法参数创业变量；变量的值会对应地从调用命令指定的参数赋为默认值给程序参数。有默认值的参数不需要在程序调用过程中特别指定。然而，它们必须是没有默认值的合法参数的调用参数，且不能有额外的调用参数。

有一个特殊的情况允许程序有参数数量成为变量。如果最后一个合法参数的名称为args，那么调用程序时就可以包含多于定义时合法变量数量。这个情况下，所有的调用参数从args开始会合并到一个列表里（就是作为命令列表被使用）；这些合并的值被分配给本地变量args。

当包体被执行了，变量名一般会指向本地变量，当引用是会自动创建，当程序返回时会自动删除。每一个程序参数都会自动创建一个本地变量。全局变量可以使用语法::来分配。

当一个程序被调用，程序的返回值就是命令指定的返回值。如果程序没有执行一个明确的返回，那么它的返回值就是包体内最后一个命令的执行返回值。如果在执行包体时发生一个错误，那么整个程序包体返回相同的错误。

return命令的语法如下：

```
return ?-code code? ?value?
```

可选择的一对参数*–code code*允许将返回代码从TH_OK修改为其它的状态代码。整个代码必须指定为一个数值。

6.6.7. 特殊命令

TH包含了三个核心的命令用于辅助命令，它们是：

```
breakpoint args
rename oldcmd newcmd
uplevel ?level? script
```

breakpoint 命令什么也不做。它作为占位符使用，用于替代调试时的断点。

rename 命令重命名用户自定义或者内建的命令。老名称会被删除，新名称会被插入到解释器的命令表。

uplevel 命令在调用链更高一级的变量域执行一个命令。脚本参数在指定层级的变量域求值。**uplevel**命令返回求值的结果。如果level是一个正整数，那么它会给出之前执行命令的距离（提升到程序调用堆栈）。如果level省略了，那么它默认值是”1“。

举例说明，假设程序在顶层被调用，然后它调用了b，然后b调用了c。假设c调用uplevel命令。如果level是”1“或者略过了，那么命令将会在b的变量环境里执行。如果level设置为”2“那么命令会在a的变量环境里执行。如果level是”3“那么命令将会在顶层执行（即只有全局参数可访问）。

uplevel命令当执行命令时会引起程序从程序的调用链里消失。在上面的例子里，假设c调用了命令：

```
uplevel 1 {set x 43; d}
```

当d是另外一个TH程序。set命令将修改b环境里的x变量，且d会在level 3执行。如果它回来执行命令

```
uplevel {set x 42}
```

那么set命令将会在b的环境修改同样的参数x：在d执行时程序c不会出现在调用链里。

6.7. 使用字符串

TH提供了**string 命令**来帮助使用字符串。string命令是一个有七个子命令的独立命令，使用第一个参数定义。第一个参数除了用于指定子命令外无任何目的。如果第一个参数没有匹配任何子命令，将会抛出错误信息。

七个子命令如下：

```
string length string
string compare string1 string2
string first needle haystack ?startindex?
string last needle haystack ?startindex?
string range string first last
string repeat string count
string is alnum string
```

string length 子命令有一个参数，也就是字符。它返回字符串长度的十进制字符。当TH使用一个一位的字符编码那么字符的尺寸是字符的尺寸和比特的尺寸。

string compare 子命令执行一个逐个字符的对比，参数为string1和string2，与C语言strcmp程序一样。它返回一个十进制的值-1、0或1，取决于string1是否少于、等于或者大于string2。

string first 子命令为一个序列的字符串准确匹配参数指针的参数堆栈里进行搜索。如果发现，它返回堆栈里匹配索引的第一个字符的十进制值。如果没有发现，返回-1。，可选的正整数参数指定了开始搜索的位置；默认值为”0“，也就是堆栈里的第一个字符。

The **string last 子命令**为一个序列的字符串准确匹配参数指针的参数堆栈里进行搜索。如果发现，它返回堆栈里匹配索引的最后一个字符的十进制值。如果没有发现，返回-1。，可选的正整数参数指定了开始搜索的位置；默认值为”0“，也就是堆栈里的第一个字符。

string range 子命令返回给定参数的连续字符串的范围，从第一个字符索引直到最后一个字符索引。如果第一个小于零那么它会被当做零进行处理，如果最后一个大于或者等于字符的长度，那么它会被当做结束进行处理。如果第一个大于最后一个，那么将返回一个空白字符。

The **string repeat string count 子命令**返回一个根据count次数重复格式化后的字符串。参数count必须是正整数，如果count是零或者小于零那么将返回一个空字符。

string is alnum 子命令测试参数字符是否是字母或数字类的字符。比如：一个字符只是由字母或数字组成。如果字符是字母数组组成那么返回一个十进制字符值1否则为0。

6.8. 使用列表

列表list是TH的基础数据结构。一个列表简单说就是序列化的对象、数字、单词、字符或其他列表。作为实例，下面的字符串是一个含四个对象的列表。第三个对象是一个含有二个对象的子列表：

```
{first second {a b} fourth}
```

TH有三个核心命令来使用列表：

```
list ?arg1 ?arg2? ...?
lindex list index
llength list
```

命令**list command**返回一个有所有参数组成的列表。括号和反斜杠可根据需要添加，那么命令lindex可能在需要将原始参数进行提取时使用。比如，命令：

```
list a b {c d e} {f {g h}}
```

将返回：

```
a b {c d e} {f {g h}}
```

命令**lindex command**参数列表作为TH列表进行处理并返回元素索引的数值。参数index必须为整数值，0表示列表的第一个元素。提取元素值时，lindex命令命令遵守与TH解释器处理括号和引号、反斜杠相关的相同规则；然而，变量转义和命令转义不会触发它。如果索引值为负值或者大于等于元素的数量，将会返回一个空的字符。

命令**llength command**将参数列表作为列表处理，返回一个十进制的字符及元素的数量。

7. 下面做什么?

这本书目前已经包含了许多关于如何使用Fossil的相关功能，我期望如此，且使得你有兴趣使用它了。问题"下面做什么"现在出现了。首先到Fossil网站http://www.fossil-scm.org/。在那里你可以查看Wiki链接然后显示所有Wiki页面。那里列出了很多深入讲解的帖子。如果你任然无法得到帮助，你可以加入Fossil的邮件列表(见Wiki 链接)查看相关存档或者直接提问。我发现那里非常有帮助且我的提问很快被答复。

在邮件列表里你将会看到很长的关于Fossil更改的讨论，有些会非常快地被接受并且在几个小时以后就会出现在Fossil的源代码里。其它引起长篇 (特别是关于Wiki的讨论)和阅读关于赞成修改建议的或弊端的非常有趣。

Fossil是一个持续进化的程序，但是如果你得到了所有你想要的功能的版本，你可以一直使用这个版本。换得新的版本时非常简单，只要在你当前仓库执行一个**rebuild**命令。开发者们很小心地实现它的基础结构，所以转换版本是很简单且安全的。

最后如果你要对这个项目贡献代码，那么有很多事情可以做(在Wiki里查看To Do List)。

List of Figures

2.1. 创建仓库 6
2.2. 打开仓库 & 检查 6
2.3. 初始文件添加 7
2.4. 初始提交 7
2.5. 开启网站服务器 8
2.6. 初始化网页服务器页面 8
2.7. 初始化配置 9
2.8. 用户配置 9
2.9. 超级用户设置 10
2.10. 项目状态 11
2.11. 更新新文件 11
2.12. 时间线 12
2.13. 时间线细节 13
2.14. 初始化标签窗口 14
2.15. 标签表单 15
2.16. 查看一个标签 16
2.17. 打开标签后的标签清单 16
2.18. 检入解决问题的标签 17
2.19. 完成Ticket标签 18
2.20. 空白的主页 19
2.21. Wiki控制 19
2.22. Wiki格式化 20
2.23. 可编辑的主页 20
2.24. 初始的主页 21

3.1. 自托管的Fossil仓库 23
3.2. Fossil CGI脚本 24
3.3. My Fossil CGI脚本 25
3.4. 复制脚本到指定位置 25
3.5. 网络访问Fossil CGI托管的网站 25
3.6. 新编辑用户 26
3.7. Clone命令 27
3.8. 克隆仓库的检入 28
3.9. 更新动作 29
3.10. 更新后的时间线 29
3.11. 作为分支提交 30
3.12. 不发行的更新 31

3.13. 小范围的不发行更新 . . . 31
3.14. 当前的时间线 . . . 32
3.15. 更新fossilbook.pdf . . . 32
3.16. 命令窗口的强制提交 . . . 33
3.17. 已分支的时间线 . . . 33
3.18. Fossil合并 . . . 33
3.19. 文本的不同之处 . . . 34
3.20. 合并的时间线 . . . 34

4.1. Marilyn的工作 . . . 36
4.2. Jim的提交意图 . . . 36
4.3. 强制的提交 . . . 37
4.4. 仓库分叉Fork . . . 37
4.5. 合并操作 . . . 38
4.6. 合并后的提交 . . . 38
4.7. 合并后的时间线 . . . 39
4.8. Marilyn的更新 . . . 39
4.9. Jim的检入企图 . . . 40
4.10. 更新演练 . . . 41
4.11. 合并后的仓库 . . . 41
4.12. 检查代码状态 . . . 42
4.13. 分支命令 . . . 43
4.14. 分支时间线 . . . 43
4.15. 设置分支在时间线上的颜色 . . . 44
4.16. 检出VER_1.0 . . . 45
4.17. VER_2.0检出 . . . 45
4.18. 修复二个分支的警告 . . . 46
4.19. 修正二个分支 . . . 47

5.1. 运行Help命令 . . . 49
5.2. 具体命令的帮助 . . . 50
5.3. add命令 . . . 50
5.4. rm命令 . . . 50
5.5. rename命令 . . . 51
5.6. status命令 . . . 51
5.7. changes命令 . . . 52
5.8. extra命令 . . . 52
5.9. revert命令 . . . 53
5.10. update命令 . . . 53
5.11. checkout或co命令 . . . 54
5.12. undo命令 . . . 54
5.14. gdiff命令 . . . 55
5.13. diff命令 . . . 55
5.15. ui命令 . . . 56
5.16. server命令 . . . 56

5.17. commit命令 57
5.18. new命令 58
5.19. clone命令 58
5.20. open命令 59
5.21. close命令 59
5.22. version命令 59
5.23. rebuild命令 60
5.24. all命令 60
5.25. push命令 61
5.26. pull命令 61
5.27. sync命令 62
5.28. clean命令 62
5.29. branch命令 63
5.30. merge命令 63
5.31. tag命令 64
5.32. settings命令 65
5.33. zip命令 66
5.34. user命令 66
5.35. finfo detail 67
5.36. timeline命令 68
5.37. wiki命令 68
5.38. scrub命令 69
5.39. search命令 69
5.40. sha1sum命令 70
5.41. rstats命令 70
5.42. configuration命令 71
5.43. descendants命令 71

Bibliography

[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.

A. 修订记录

使用如下表格来追踪文档的版本历史，任何修改都可以添加一行来阐明所做的修改内容 .

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
2.0	FvD	Updated and matched to fossil 1.25	March 2013
2.0-cn	Lomatus	Start Chinese Translation Base on 2.0	10-Sep- 2013