1.关于Git的一些小知识

`Git` `SVN` `CVCS` `分布式版本控制系统` `集中式版本控制系统`

一贯风格，图片，不存在的，暂时没时间做。

> 感受写在前面吧。写完这篇笔记后，感觉Git原理就是很简单的一个东西。Git绝大数操作都是建立副本的（快照），然后提交操作就是把副本（快照）通过压缩算法压缩后存入数据库（在Git目录中）中。至于对比文件是否修改了，使用的就是hash值算法。而文件里面具体改变的内容是什么，那就是靠另外的一套对比算法了。所以为了提高提交的效率啊，Git并不关心单个文件细节改变，它只关注了你的单个文件是否发生了改变（对比hash值呗）。

一、浅谈版本控制的发展史

1. **最原始最简单粗暴的版本控制**
   那肯定就是你在个人电脑上通过简单的复制粘贴整个文件夹，然后文件夹名加上时间段等后缀来建立的不同版本啦。
2. **本地控制系统**
   稍微智能化了一丁点，就是有软件帮你把前后修改的单个文件做出patch（补丁）。其实这个补丁就是标识出了单个文件和上个版本对比改变了什么内容。我估计内部保存副本的方式还是暴力的复制粘贴。
3. **集中化的版本控制系统--CVCS（Centralized Version Control Systems）**
   目前大部分公司在用的就是它了，典型的就是一只小乌龟图标的SVN。
   CVCS有一个单一集中管理的服务器，保存所有文件的修订版本，而协同工作的人通过客户端连接到服务器更新或提交自己的东西。

CVCS优点：

- 团队里的人可以在一定程度上知道其他人在干什么;
- 管理员可以管理每个开发者的权限;
- 管理员管理一个CVCS服务器要比管理各个客户端的本地数据库轻松。

CVCS缺点:

- 服务器挂了的话，大家都GG思密达，有很大的丢失数据的风险。因为这种系统，只下载最新版本的快照。

4. **分布式版本控制系统--DVCS（Distributed Version Control Systems）**
   DVCS系统，客户端并不是提取最新版本的文件快照，而是把代码仓库完整镜像下来。这样的话，任何一处协同工作用的服务器挂了的话，事后可以用任何一个镜像出来的本地仓库来恢复。更加人性化的是，DVCS系统还允许你指定和若干不同远端代码仓库的人员进行交互，也就是说你可以在同一个项目中分别和不同工作小组的人相互协作。

二、Git教科书般的励志历史

> Git诞生前，在Linux系统下，开发项目普遍都用BitKeeper分布式系统。BitKeeper为Linux提供了三年的免费服务。结果三年过后宣布要收费，Linux开源社区一怒之下就决定开发属于自己的DVCS了。
> 它们指定DVCS的目标如下：
> 1. **速度（我个人理解为时间复杂度）;**
> 2. **简单设计（一个控制台几条指令是挺简单的）;**
> 3. **对非线性开发模式的强力支持;**
> 4. **完全分布式;**
> 5. **有能力高效管理类似Linux内核一样的超大规模项目（效率和数据量）。**

三、Git的一些基基基基础

1. 普通的CVS系统都关心单个文件具体的差异，而Git则关注单个文件到底有没变（个人理解还是时间复杂度问题）；
2. Git近乎所有的操作都是本地执行的。包括离线提交更新后，等到有网络了，它会上传到远程仓库。

3. Git能时刻保持数据完整性。Git使用SHA-1算法计算数据的校验和。Git的工作完全依赖于哈希值，所有保存在Git数据库中的东西都是用这哈希值进行索引的，而不是靠文件名。所以网络传输或硬盘损坏丢失了数据，Git都能知道（就是类似于暴雪公司那套神奇文件hash算法）。

4. Git多数操作仅添加数据。常用Git操作绝大部分只是往数据库里添加数据。Git完美避免了“删除操作”等不可逆操作时无法回退重现历史版本的问题。个人理解，当用户执行删除文件并且提交后，Git只是在它数据库的目录数据结构上把该文件节点给删除了，真正的文件还压缩在数据库中。所以要恢复，毫不费劲。只是改下节点值而已。

5. Git文件的三种状态：

   - 已提交（committed）：文件已经被安全保存在了本地数据库中了；
   - 已修改（modified）：文件已被修改，但未提交保存；
   - 已暂存（staged）：把已修改的文件放在下次提交时要保存的清单中。

   所以Git文件流转的三个区域是：**工作目录、暂存区域和本地仓库。**
   工作目录： 从项目中取出某个版本的所有文件和目录，用以开始后续工作的目录就叫工作目录。这个目录实际上是从Git目录的压缩数据库中提取出来的数据，接下来的工作就是对这些文件进行编辑。
   暂存区域：暂存区域其实就是一个简单的文件，一般都放在Git目录中；
   本地仓库：每个项目都有一个Git目录（通过`git clone`出来的话，就是其中的`.git`目录；如果是`git clone --bare`的话，新建的目录就是Git的目录）。这个目录用来保存元数据以及对象数据库的地方，每次克隆的镜像仓库，拷贝的就是这个目录里面的数据了。

   基本的Git工作流程如下：
   1. 在工作目录中修改某些文件
   2. 对修改后的文件进行快照，然后保存到暂存区域。
   3. 提交更新，将保存在暂存区域的文件快照永久转储到Git目录中，也就是压缩后放到数据库中。

6. 安装Git的具体教程，自己看吧。
   ProGit

7. Git初次运行时的配置：
   Linux现在接触少，就不说它的配置方式了，以后再说；
   Windows的话，Git会寻找主目录（$HOME）下的`gitconfig`文件。此外，Git还会尝试寻找`/etc/gitconfig`文件，就是Git的安装目录啦。

8. Git安装后的一些配置，直接打开`Git Bash.exe`往里面写配置吧。

```
//用户信息配置：
$ git config --global user.name "Flew"
$ git config --global user.email DontTellU@163.com
```

```
//文本编辑器配置:还是用vim吧
$ git config --global merge.tool Vim
```

```
//差异分析工具：指定在解决合并冲突时使用哪种差异分析工具
$ git config --global merge.tool vimdiff
```

```
//查看配置信息（简单粗暴）
$ git config --list
```

```
//获取帮助：
$ git help <verb>
$ git help config
```

> 参考文献：
> ProGit多国语言版:**神Git教材，一点就通，简单粗暴，不夹废话，都是精髓。**