不管你是从前面的章节直接跳到了本章，还是读完了其余各章一直到这，你都将在本章见识 Git 的内部工作原理和实现方式。我个人发现学习这些内容对于理解 Git 的用处和强大是非常重要的，不过也有人认为这些内容对于初学者来说可能难以理解且过于复杂。正因如此我把这部分内容放在最后一章，你在学习过程中可以先阅读这部分，也可以晚点阅读这部分，这完全取决于你自己。

既然已经读到这了，就让我们开始吧。首先要弄明白一点，从根本上来讲 Git 是一套内容寻址 (content-addressable) 文件系统，在此之上提供了一个 VCS 用户界面。马上你就会学到这意味着什么。

早期的 Git (主要是 1.5 之前版本) 的用户界面要比现在复杂得多，这是因为它更侧重于成为文件系统而不是一套更精致的 VCS 。最近几年改进了 UI 从而使它跟其他任何系统一样清晰易用。即便如此，还是经常会有一些陈腔滥调提到早期 Git 的 UI 复杂又难学。

内容寻址文件系统层相当酷，在本章中我会先讲解这部分。随后你会学到传输机制和最终要使用的各种库管理任务。

本书讲解了使用 checkout, branch, remote 等共约 30 个 Git 命令。然而由于 Git 一开始被设计成供 VCS 使用的工具集而不是一整套用户友好的 VCS，它还包含了许多底层命令，这些命令用于以 UNIX 风格使用或由脚本调用。这些命令一般被称为 "plumbing" 命令（底层命令），其他的更友好的命令则被称为 "porcelain" 命令（高层命令）。

本书前八章主要专门讨论高层命令。本章将主要讨论底层命令以理解 Git 的内部工作机制、演示 Git 如何及为何要以这种方式工作。这些命令主要不是用来从命令行手工使用的，更多的是用来为其他工具和自定义脚本服务的。

当你在一个新目录或已有目录内执行 git init 时，Git 会创建一个 .git 目录，几乎所有 Git 存储和操作的内容都位于该目录下。如果你要备份或复制一个库，基本上将这一目录拷贝至其他地方就可以了。本章基本上都讨论该目录下的内容。该目录结构如下：

$ ls

HEAD

branches/

config

description

hooks/

index

info/

objects/

refs/

该目录下有可能还有其他文件，但这是一个全新的 git init 生成的库，所以默认情况下这些就是你能看到的结构。新版本的 Git 不再使用 branches 目录，description 文件仅供 GitWeb 程序使用，所以不用关心这些内容。config 文件包含了项目特有的配置选项，info 目录保存了一份不希望在 .gitignore 文件中管理的忽略模式 (ignored patterns) 的全局可执行文件。hooks 目录保存了第七章详细介绍了的客户端或服务端钩子脚本。

另外还有四个重要的文件或目录：HEAD 及 index 文件，objects 及 refs 目录。这些是 Git 的核心部分。objects 目录存储所有数据内容，refs 目录存储指向数据 (分支) 的提交对象的指针，HEAD 文件指向当前分支，index 文件保存了暂存区域信息。马上你将详细了解 Git 是如何操纵这些内容的。