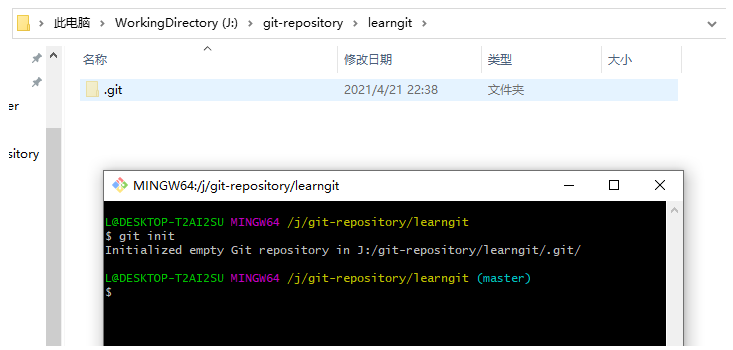
**Git原理 - 详解.git目录（一）**

**1、.git目录介绍**

我们先在本地电脑中创建一个Git本地版本库。

1. 在自己指定的目录中新建一个文件夹。
2. 进入该文件夹，右键启动Git Bash
3. 输入git init 命令来初始化一个Git本地版本库。

如下：



我们可以看到，一个刚刚初始化好的Git本地版本库中，只有一个.git目录，此刻这也叫做Git的裸仓库。

|  |
| --- |
| 说明：   * 当在一个新目录或已有目录执行 git init 时，Git 会创建一个 .git 目录。 这个目录包含了几乎所有 Git 存储和操作的东西。 * .git 目录中的这些文件是 Git 仓库的骨干，也是Git的最核心的东西。 * 如若想备份或复制一个版本库，只需把这个目录拷贝至另一处即可。 |

**.git目录初始内容如下**：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) total 7 -rw-r--r-- 1 L 197121 130 4月 21 22:38 config -rw-r--r-- 1 L 197121 73 4月 21 22:38 description -rw-r--r-- 1 L 197121 23 4月 21 22:38 HEAD drwxr-xr-x 1 L 197121 0 4月 21 22:38 hooks/ drwxr-xr-x 1 L 197121 0 4月 21 22:38 info/ drwxr-xr-x 1 L 197121 0 4月 21 22:38 objects/ drwxr-xr-x 1 L 197121 0 4月 21 22:38 refs/ |

今天我们到.git目录中去看一看。

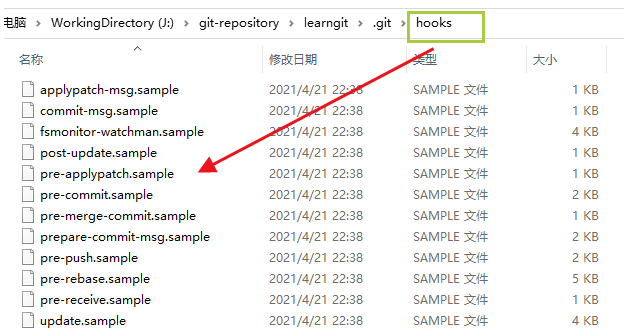
**2、hooks目录 (钩子)**

**（1）hooks目录介绍**

和其它版本控制系统一样，Git 能在特定的重要动作发生时，触发自定义脚本（钩子脚本）。

钩子分为：客户端的钩子和服务器端的钩子。客户端钩子由提交和合并这样的操作所调用，而服务器端钩子作用于接收被推送的提交这样的联网操作。

钩子都被存储在.git目录下的 hooks 子目录中。 如下图：



当你用 git init 命令初始化一个新版本库时，Git 默认会在这个目录中放置一些示例脚本。这些脚本都是 shell 脚本，其中一些还混杂了 Perl 代码。你可以使用任何你熟悉的语言编写Git钩子脚本，如Ruby 或 Python等编写的可执行脚本，都可以正常使用。

将编写好的可执行脚本（不带扩展名），放入 .git 目录下的 hooks 子目录中，即可激活该钩子脚本。

**（2）钩子脚本举例说明**

1. pre-commit 钩子：在创建提交信息前运行，它用于检查即将提交的快照。例如，检查是否有所遗漏，确保测试运行，以及核查代码。
2. 如果该钩子以非零值退出，Git 将放弃此次提交，不过你可以用 git commit --no-verify 来绕过这个环节。
3. 你可以利用该钩子，来检查代码风格是否一致、尾随空白字符是否存在，或新方法的文档是否适当等操作。
4. commit-msg 钩子：接收一个参数，此参数存有当前提交信息的临时文件的路径。 如果该钩子脚本以非零值退出，Git 将放弃提交，因此，可以用来在提交通过前验证项目状态或提交信息。
5. post-commit 钩子：在整个提交过程完成后运行。 它不接收任何参数，但你可以很容易地通过运行 git log -1 HEAD 来获得最后一次的提交信息。 该钩子一般用于通知之类的事情。

这里只简单介绍三个hooks目录中的钩子脚本，如果想查看更多钩子示例脚本说明，可以查看[https://git-scm.com/book/zh/v2/%E8%87%AA%E5%AE%9A%E4%B9%89-Git-Git-%E9%92%A9%E5%AD%90#\_git\_hooks](https://links.jianshu.com/go?to=https%3A%2F%2Fgit-scm.com%2Fbook%2Fzh%2Fv2%2F%25E8%2587%25AA%25E5%25AE%259A%25E4%25B9%2589-Git-Git-%25E9%2592%25A9%25E5%25AD%2590%23_git_hooks)

**3、info目录**

info 目录包含一个全局性排除（global exclude）文件， 用以放置那些不希望被记录在 .gitignore 文件中的忽略模式（ignored patterns）

（具体用法，可以看前面**Git忽略**相关的文章）

**4、description文件**

description文件：仓库的描述信息。

description文件内容如下：

|  |
| --- |
| Bash $ cat description Unnamed repository; edit this file 'description' to name the repository. |

翻译：未命名的存储库； 编辑此文件“description”，以命名存储库。

说明：该文件仅供 GitWeb (Github 的一种前身) 程序使用，我们无需关心。

**5、config文件**

config 文件包含项目特有的配置选项。

Git配置分为三个级别：

1. 系统级别：system。
2. 全局级别（用户级别）：global。
3. 本地级别：local。

本地级别的配置信息，就记录在config 文件中，使用git config --local或git config 不加任何参数命令进行配置。

（具体应用，可以看前面**Git配置签名**相关的文章）

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/23012e3f9519 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。