自从看了欧阳助教的 Intro-to-git 视频，小蓝鲨陷入了对 git 的沉迷无法自拔。恰好，最近小蓝鲨学习了一些文件系统的一些知识，于是小蓝鲨决定在寒假实现一个类似于 git 的文件系统:"git--"。由于小蓝鲨水平不够，git--仅需要完成 git 的部分命令。

git-- 是一个基于内存的文件系统，其使用内存中的结构来保存和管理用户的文件数据。下图展示了 git-- 的工作流程和 git-- 保存数据的基本方式。在使用 git-- 时，用户发送请求给 git--。git-- 会解析用户的请求，在内存数据结构中进行操作，完成用户请求。

git--中的一些基本概念：

• 文件：与普通文件系统中的文件概念相同，git-- 中的文件可以保存数据。git-- 中的文件支持读取、写入、删除。写入一个不存在（或此前已被删除）的文件会自动创建该文件

• 文件名：每个文件拥有一个文件名，为一个长度不超过 128 的字符串。文件名的内容仅包含大小写英文字母（A~Z 和 a~z）和数字（0~9）。小蓝鲨认为 git-- 的用户都非常善良，所以他们能够保证每个 git-- 命令中的文件名均满足上述规定，git-- 在实现的时候无需进行检查。

• 暗文件：为了表示文件的删除，git-- 中引入了暗文件的概念。对于一个名为 filename 的文件，其暗文件名称为 - filename（即在文件名前增加了一个负号 -）。由于在用户创建的文件名中不允许出现 -，暗文件的文件名与普通文件的文件名并不会混淆。暗文件的文件大小为 0，其中不可保存数据。其仅表示名为 filename 的文件被删除。一个文件与其暗文件不应同时出现在同一个提交中，也不应同时出现在暂存区中（关于提交和暂存区的概念请看下面两条）。如下图中的 - file2 为一个暗文件，表示对文件 file2 的删除。

• 暂存区：用户所有的修改，包括文件写入、删除等，在未提交时，均保存在暂存区（即下图中的 uncommitted 结构，包括虚线椭圆及其右侧的文件）。其中文件的写入（包括创建）以文件的形式保存，而文件的删除以暗文件的形式保存。

• 提交：与在 Git 中类似，一个提交表示 git-- 的一个历史状态。一般来说，提交由 commit 命令创建，git-- 将当前暂存区中的所有修改保存下来，并赋予一个唯一的提交名，成为一个提交。提交名的命名要求与文件名相同，且无需进行格式检查。除了 commit 命令外，用户还可以通过 merge 命令创建一个提交。通过 merge 命令创建的提交将两个现有提交进行合并。提交的创建在后文中有具体描述。除了 git-- 中的第一个提交不存在父提交之外，其余每个提交拥有一个父提交（由 commit 命令创建）或者两个父提交（由 merge 命令创建）。如下图中的 cmt1 表示一个名为 cmt1 的提交。

• HEAD：与 Git 中类似，HEAD 表示当前的头部，指向头部提交。用户当前能够访问哪些文件，以及文件的内容，取决于当前缓存区中的内容以及当前头部所指向的提交。

• git-- 命令：用户使用 git-- 命令对 git-- 的内容进行操作。命令只能通过标准输入传递给 git--，除了 write 命令占据两行之外，其他 git-- 均只占一行（即以换行符结尾）。

git-- 的存储结构

git—中保存了一个头部，一个暂存区结构，和若干个提交结构。

• **头部（HEAD）**：git-- 中有且仅有一个头部，为指向当前头部提交的一个指针或者引用，当 git—中不存在任何提交时，头部为空。

• **暂存区结构（uncommitted）**：git--中有且仅有一个暂存区结构。暂存区结构中包含了所有还未提交的修改。在此结构中，应该保存所有被修改（包括创建）文件的名称、大小和内容。对于删除的文件，该结构中应当保存对应的暗文件的信息。注意暗文件的大小为 0，不保存数据，因此只有暗文件的文件名是有意义的信息。

• **提交结构**：git-- 中可以有零个或者多个提交结构，保存在元数据结构之中。暂存区结构中的内容在提交后，变为提交结构。一旦生成，提交结构中的内容是不可修改的。

初始状态 一个刚刚被创建出来的 git--文件系统只包括一个空头部和一个空的暂存区结构（即其中没有任何文件或暗文件）。