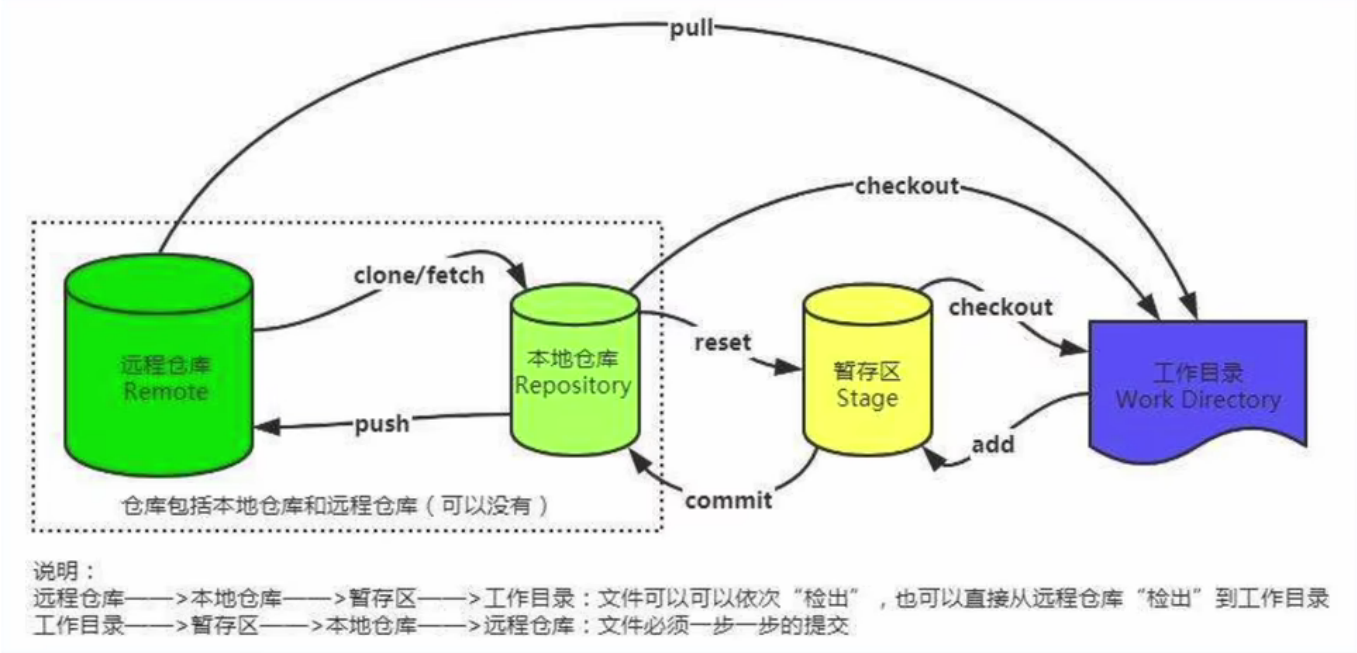
工作流程：



说明：
远程仓库——>本地仓库——>暂存区——>工作目录：文件可以可以依次“检出”，也可以直接从远程仓库“检出”到工作目录
工作目录——>暂存区——>本地仓库——>远程仓库：文件必须一步一步的提交

创建

创建本地仓库

我们可以将任意一个文件夹作为一个本地仓库，输入：

```
git init
```

输入后，会自动生成一个 .git 目录，注意这个目录是一个隐藏目录，而当前目录就是我们的工作目录。

创建成功后，我们可以查看一下当前的一个状态，输入：

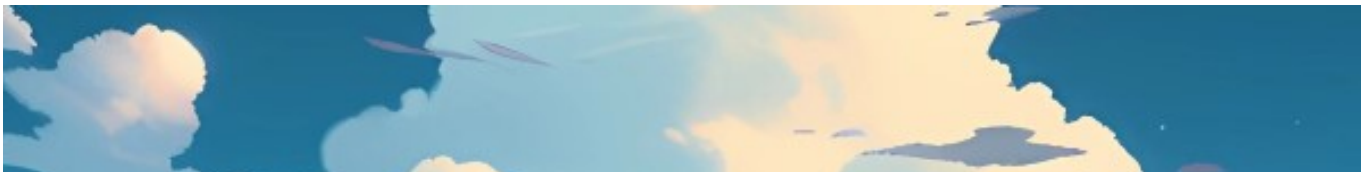
```
git status
```

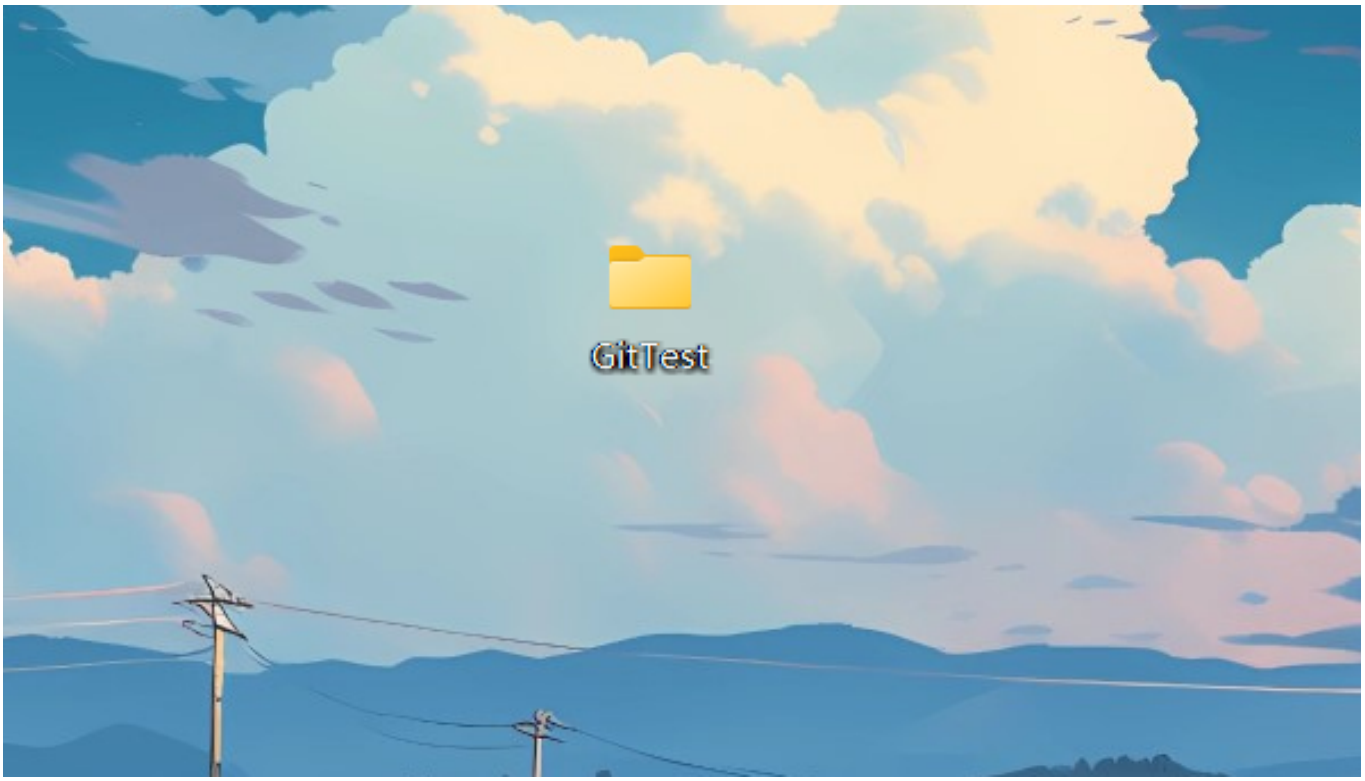
如果已经成功配置为Git本地仓库，那么输入后可以看到：

```
On branch master

No commits yet
```

这表示我们还没有向仓库中提交任何内容，也就是一个空的状态。

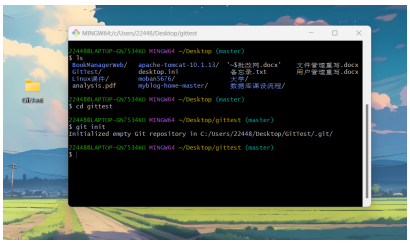
先创建一个文件夹：



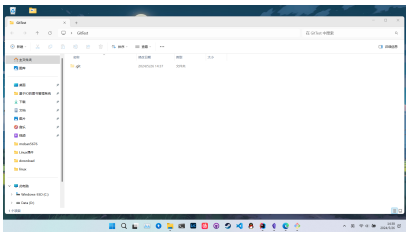


在该文件夹路径下开打**gitbash**（可以到文件夹里面打开**gitbash**，也可以先打开**gitbash**,路径改到文件夹）：输入**git init** *将这个文件夹初始化为**git**仓库*

(这里选择第二种方式:)



可以看到里面多了一个**.git**的文件

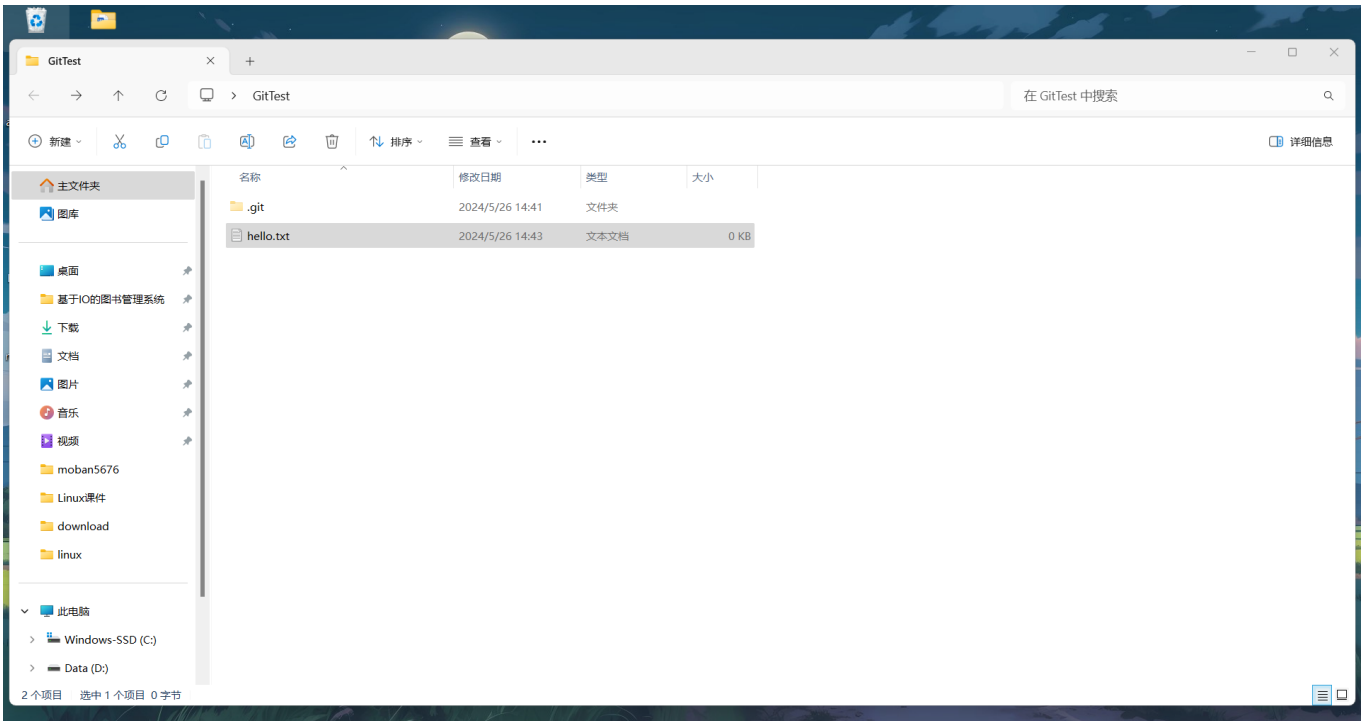


输入：**git status** 查看状态：

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

添加

我们先在初始化好的仓库里添加一个**hello.txt**的文本，里面写好**hello world**

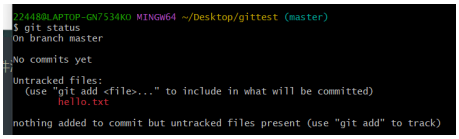


写完之后，我们就可以把这个**txt**文件添加到版本控制里面，但是他现在处于一个未追踪状态，就是**git**不会去记录他的版本迭代信息的

我们先输入：

```git
git status
```

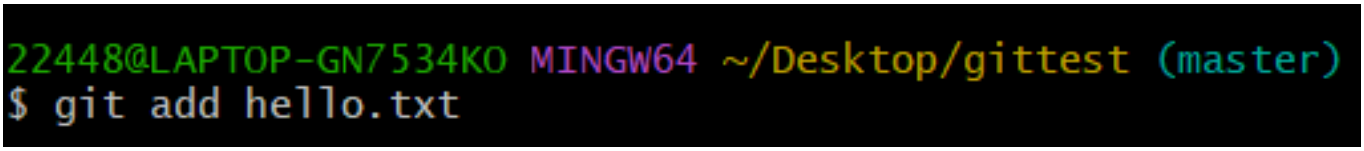
显示： *出现了未追踪文件*



这时候，我们输入：

将*添加*到暂存区，设定为追踪状态

```git
git add 文件名.后缀  #也可以使用 git add .  就是一次性将目录下的内容全部添加
```



这时候，我们再来输入：

```git
git status
```

出现了等待被提交，说明，**hello.txt**文件已经被添加到了暂存区里面了 

提交

*接着我们来尝试将其提交到**Git**本地仓库中，注意需要输入提交的描述以便后续查看，比如你这次提交修改了或是新增了哪些内容：

```git
git commit -m '描述'
```

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git commit -m '初始化提交'
[master (root-commit) 3e59e8d] 初始化提交
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
```

*接着我们可以查看我们的提交记录：

```git
后面的是参数，可以多个参数写一起
git log
git log --graph   #以图形的方式显示，如果出现很多的那些分支就可以使用
git log --oneline #如果提交的东西很多，可以试一下这个，可以显示成一行
git log --all     #全部提交
```

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log
commit 3e59e8dfdf161db9b8090fb219298141b82ad6c1 (HEAD -> master)
Author: 努力学习java的耶 <133225229+123asdasdnk@users.noreply.github.com>
Date:   Sun May 26 15:01:48 2024 +0800

    初始化提交
```

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --oneline
3e59e8d (HEAD -> master) 初始化提交
```

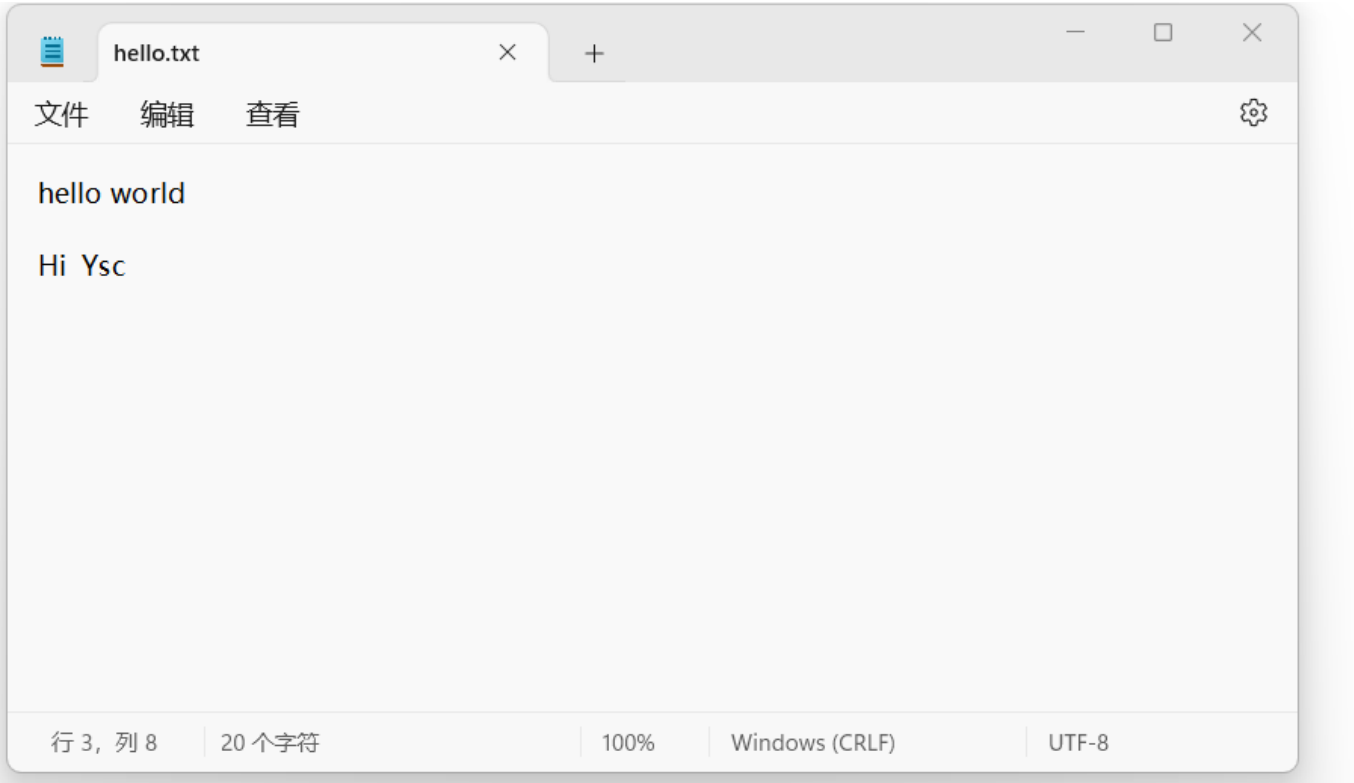
```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --graph
* commit 3e59e8dfdf161db9b8090fb219298141b82ad6c1 (HEAD -> master)
  Author: 努力学习java的耶 <133225229+123asdasdnk@users.noreply.github.com>
  Date:   Sun May 26 15:01:48 2024 +0800

      初始化提交

22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --oneline --graph
* 3e59e8d (HEAD -> master) 初始化提交

22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --oneline --graph --all
* 3e59e8d (HEAD -> master) 初始化提交
```

至此，我们已经把我们的第一个版本的文件**hello.txt**提交到本体仓库了

*好！这时候，如果我想搞第二个版本，我们先修改一下**hello.txt**里面的内容*

我们多加一个：**Hi Ysc**

这时候，我们这个**hello.txt**文件更新了，就是第二个版本了，相当于第一版本多增加了**Hi Ysc**



这时候：我们来查看一下状态

```git
git status
```

是不是显示了一个文件被修改了，是不是之前被追踪了，可以看现在的状态了

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

这时候，我们可以继续先添加到暂存区，然后提交：

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git add hello.txt

22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git commit -m '修改后提交'
[master 9c697e7] 修改后提交
 1 file changed, 3 insertions(+), 1 deletion(-)
```

查看提交记录：

可以看到，已经有了两次提交了，*并且HEAD在哪，哪一个就是最新的版本*

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --oneline --graph --all
* 9c697e7 (HEAD -> master) 修改后提交
* 3e59e8d 初始化提交
```

我们还可以查看最近一次变更的详细内容：

```GIT
git show [也可以加上commit ID 查看指定的提交记录]
```

git show:
可以看到，
老的为：**hello world**
新的为：
hello world

Hi Ysc 

git show 9c697e7 

现在我们再来查看一下状态：已经是清空状态了

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git status
On branch master
nothing to commit, working tree clean
```

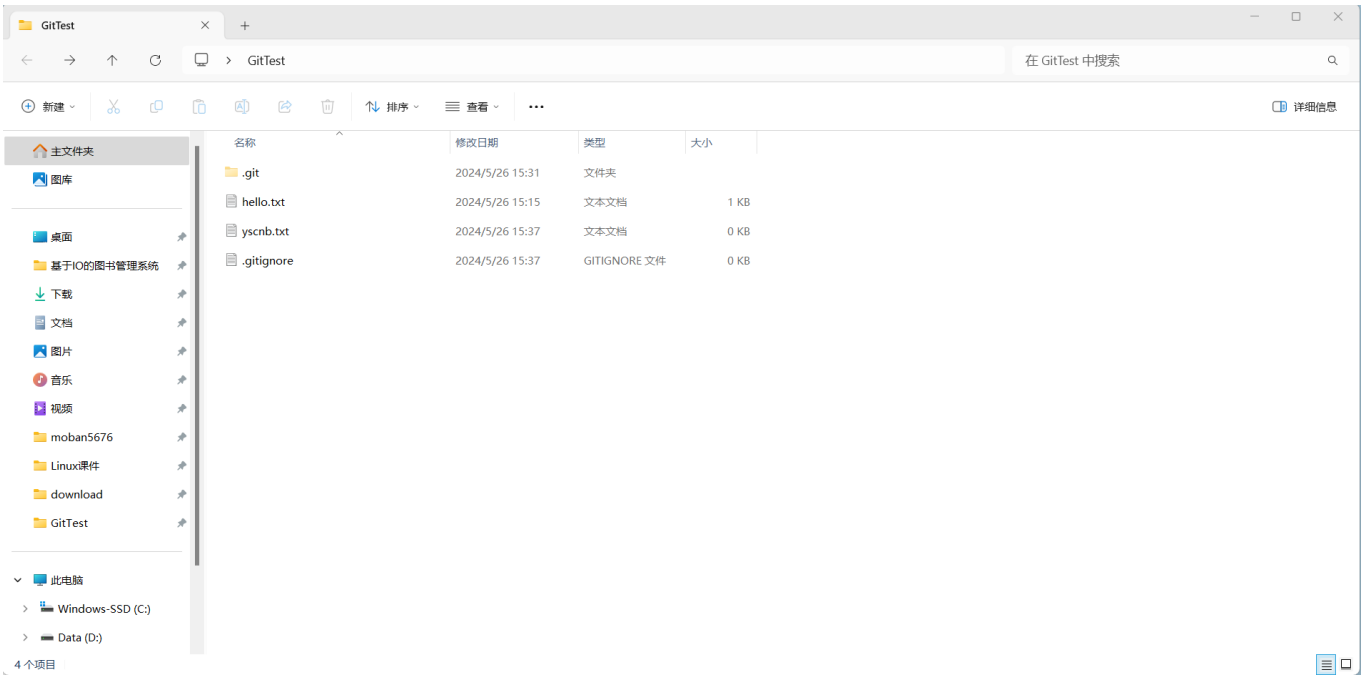
我们还可以创建一个.gitignore文件来确定一个文件忽略列表，如果忽略列表中的文件存在且部署被追踪状态，那么git不会对其进行任何检查：

我们可以创建一个 .gitignore 文件来确定一个文件忽略列表，如果忽略列表中的文件存在且不是被追踪状态，那么git不会对其进行任何检查：

```
# 这样就会匹配所有以txt结尾的文件
*.txt
# 虽然上面排除了所有txt结尾的文件，但是这个不排除
!666.txt
# 也可以直接指定一个文件夹，文件夹下的所有文件将全部忽略
test/
# 目录中所有以txt结尾的文件，但不包括子目录
xxx/*.txt
# 目录中所有以txt结尾的文件，包括子目录
xxx/**/*.txt
```

创建后，我们来看看是否还会检测到我们忽略的文件。

演示：我们另外创一个**yscnb.txt** 和 **.gitignore**文件



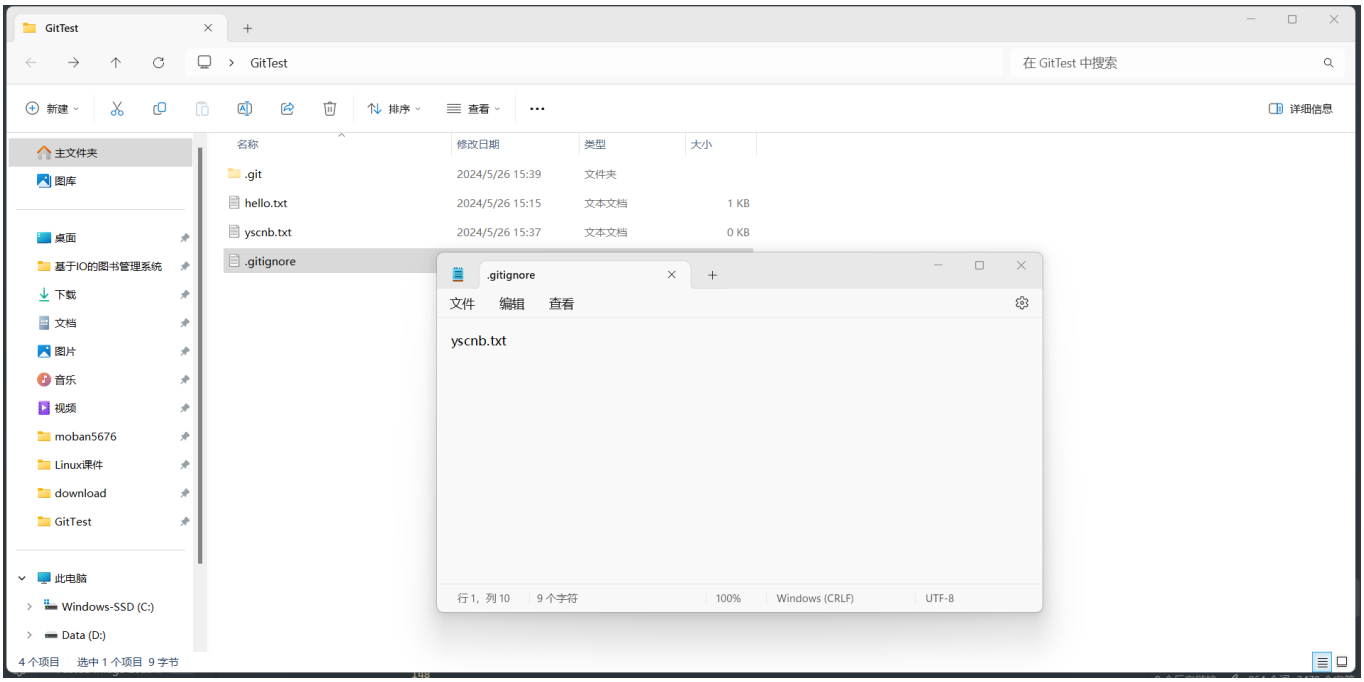
我们现在来看看状态：

新增了两个文件：`.gitignore` 和`yscnb.txt`

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        yscnb.txt

nothing added to commit but untracked files present (use "git add" to track)
```

如果这时候，我希望吧`yscnb.txt`文件给忽略掉，那么我们可以这么做：

把 文件名.扩展名 直接写入到`.gitignore`文件中就可以了：



我们现在再来看看状态：

发现`yscnb.txt`文件是不是没有了,因为`.gitignore`文件生效了，会先看`.gitignore`文件里面的东西，在里面的会被忽略

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

`.gitignore`有很多的匹配规则：

```
# 这样就会匹配所有以.txt结尾的文件
```

```
*.txt
# 虽然上面排除了所有txt结尾的文件，但是这个不排除
!666.txt
# 也可以直接指定一个文件夹，文件夹下的所有文件将全部忽略
test/
# 目录中所有以txt结尾的文件，但不包括子目录
xxx/*.txt
# 目录中所有以txt结尾的文件，包括子目录
xxx/**/*.txt
```

现在我们来把`.gitignore`也提交了：

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git add .gitignore

22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git commit -m '创建.gitignore'
[master 67cb75e] 创建.gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
```

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git log --oneline --graph --all
* 67cb75e (HEAD -> master) 创建.gitignore
* 9c697e7 修改后提交
* 3e59e8d 初始化提交
```

回滚

现在，我们想要回退到过去的版本时，就可以进行回滚操作，执行后，可以将工作空间的内容恢复到指定提交的状态：

```GIT
git reset --hard commitID  #这里的hard表示，不保留所有的文件，当然也有软
```

我们回到第一次提交的状态：

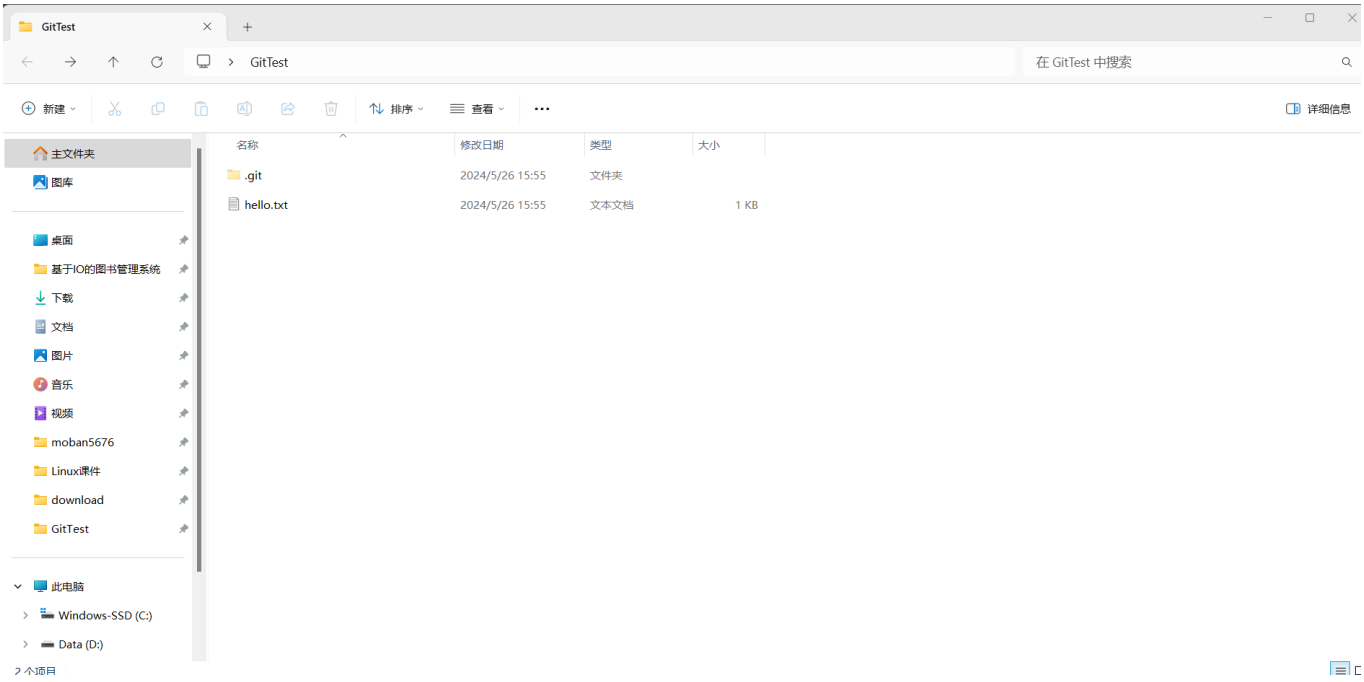
```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git reset --hard 3e59e8d
HEAD is now at 3e59e8d 初始化提交
```

查看日志：

发现确实回到了第一次提交的时候了

并且.gitignore也没有了，hello.txt里面的内容回到最初(这里yscnb.txt还有是因为我们之前把其添加到.gitignore文件里面了，被忽略了,不用管，可以删了)

删了yscnb.txt：



那么要是现在我又想回去呢，（并且我现在恰好把之前的commitID的消息删除了）?就是回到最新版本，我们可以通过查看所有分支的所有操作记录：

```GIT
git reflog
```

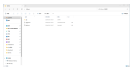
这样就可以*找到之前的commitID*,再次重置即可

这个命令可以，把所有进行过的提交或回滚或分支都记录下来，不会丢失
这样就可以查看之前的东西了

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git reflog
3e59e8d (HEAD -> master) HEAD@{0}: reset: moving to 3e59e8d
67cb75e HEAD@{1}: commit: 创建.gitignore
9c697e7 HEAD@{2}: commit: 修改后提交
3e59e8d (HEAD -> master) HEAD@{3}: commit (initial): 初始化提交
```

现在，我们知道了最新版本的**commitID,**我们再次回滚**(reset)**：
这时候，我们就重新回到了最新版本

```
22448@LAPTOP-GN7534KO MINGW64 ~/Desktop/gittest (master)
$ git reset --hard 67cb75e
HEAD is now at 67cb75e 创建.gitignore
```

查看文件夹，发现确实回来了   

查看日志：
发现，**HEAD**又到了最新版本了

所以说**reset**不会把之前的删除，只是隐藏了

另外再提一嘴：

当我们要提交的时候，是不是要先：
git add 文件名.后缀
然后进行提交：
git commit -m '描述'

这样是不是要两步，其实只需要一步就可以了：
直接**commit:**
git commit -m '描述'

注意：

这种方式只能适用于：*文件之前已经被提交过的*，如果文件是新创建的还没有被提交过的，是不能用这种方式，必须老老实实先**add** 再**commmit**