

# 系统开发工具基础第一次实验报告

姓名：杨子程 学号：23020007145 专业：计算机科学与技术

2024 年 8 月 24 日

## 目录

<b>1</b>	<b>关于Git的使用</b>	<b>3</b>
1.1	创建一个可使用Git的本地仓库 . . . . .	3
1.2	将本地仓库和Github作链接 . . . . .	5
1.3	将本地文件添加到Github仓库中 . . . . .	6
1.4	创建新分支并在新分支上进行操作 . . . . .	8
1.5	合并分支操作 . . . . .	9
1.6	从云端向本地的克隆操作 . . . . .	10
1.7	撤销对工作区文件的修改 . . . . .	11
1.8	撤销对缓存区的修改 . . . . .	12
1.9	撤销最新一次提交但保留暂存区记录 . . . . .	13
1.10	撤销最新一次提交且删除缓存区记录 . . . . .	14
<b>2</b>	<b>关于Latex的使用</b>	<b>15</b>
2.1	文件的基本格式和标题的生成 . . . . .	15
2.2	目录的生成 . . . . .	15
2.3	章节的生成 . . . . .	16
2.4	彩色字体的使用 . . . . .	16
2.5	自定义页码 . . . . .	17
2.6	列表（清单）的生成 . . . . .	17
2.7	表格的生成 . . . . .	18

---

2.8 图表的使用 . . . . .	19
2.9 插入各种数学公式 . . . . .	20
2.10 书写参考文献 . . . . .	21

# 1 关于Git的使用

## 1.1 创建一个可使用Git的本地仓库

首先，在本地的一个合适的目录下新建一个用于作为仓库的文件夹。然后，使用Git Bash切换到对应的文件夹，使用命令

```
git init
```

相关过程：

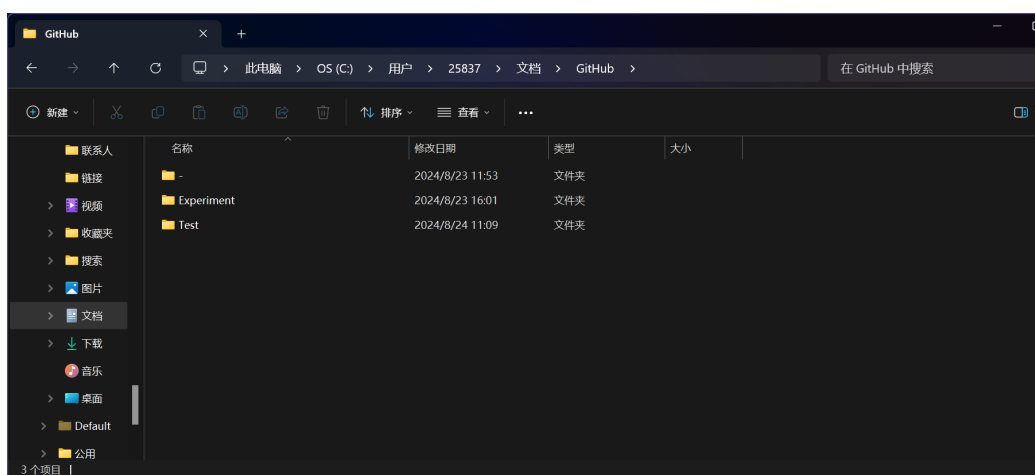


图 1: 本地文件夹

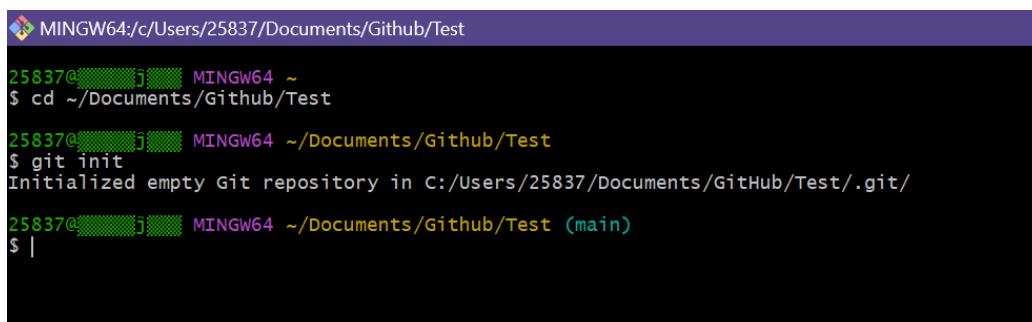


图 2: Git Bash的操作



图 3: 完成创建

## 1.2 将本地仓库和Github作链接

首先，登录到Github创建一个新的Repository，用来作为储存你新项目的云端仓库，之后使用Git Bash切换到你的本地仓库，使用命令进行链接操作。命令如下：

```
git remote add origin 你的Github上对应的Repository的网址
```

相关过程：

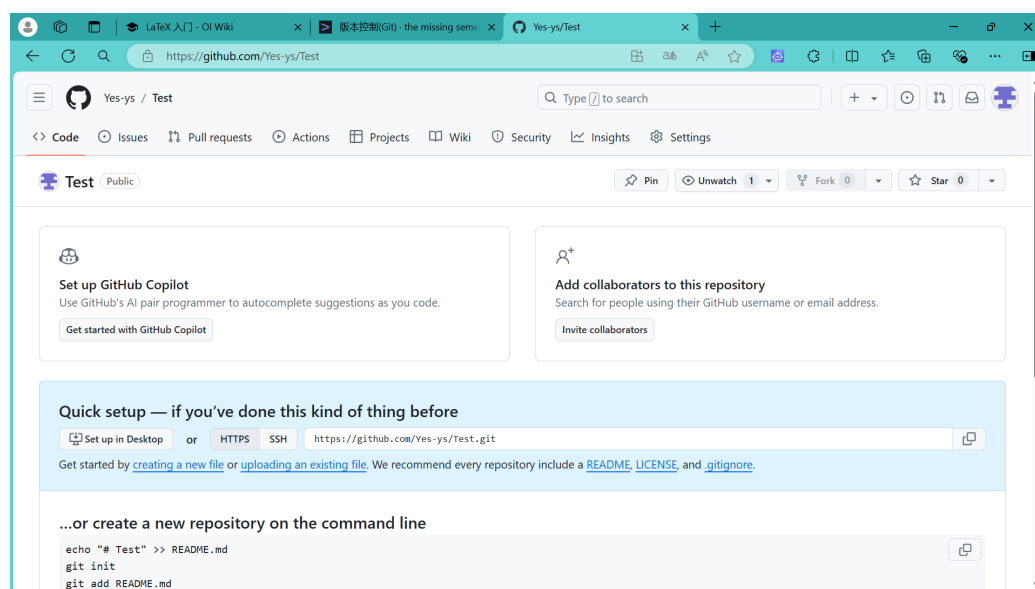


图 4: Github上的操作

```
25837@MINGW64 ~/Documents/Github/Test (main)
$ git remote add origin https://github.com/Yes-ys/Test

25837@MINGW64 ~/Documents/Github/Test (main)
$ git remote add origin https://github.com/Yes-ys/Test.git
error: remote origin already exists.

25837@MINGW64 ~/Documents/Github/Test (main)
$
```

图 5: Git Bash的操作（通过重复执行上述命令Git Bash返回的信息可以看出操作成功）

## 1.3 将本地文件添加到Github仓库中

首先，对本地文件夹的内容进行更新（此处新建了一个txt文件）。然后，将工作区的更新存入暂存区。之后，对暂存区的内容进行提交。最后，上传到Github。

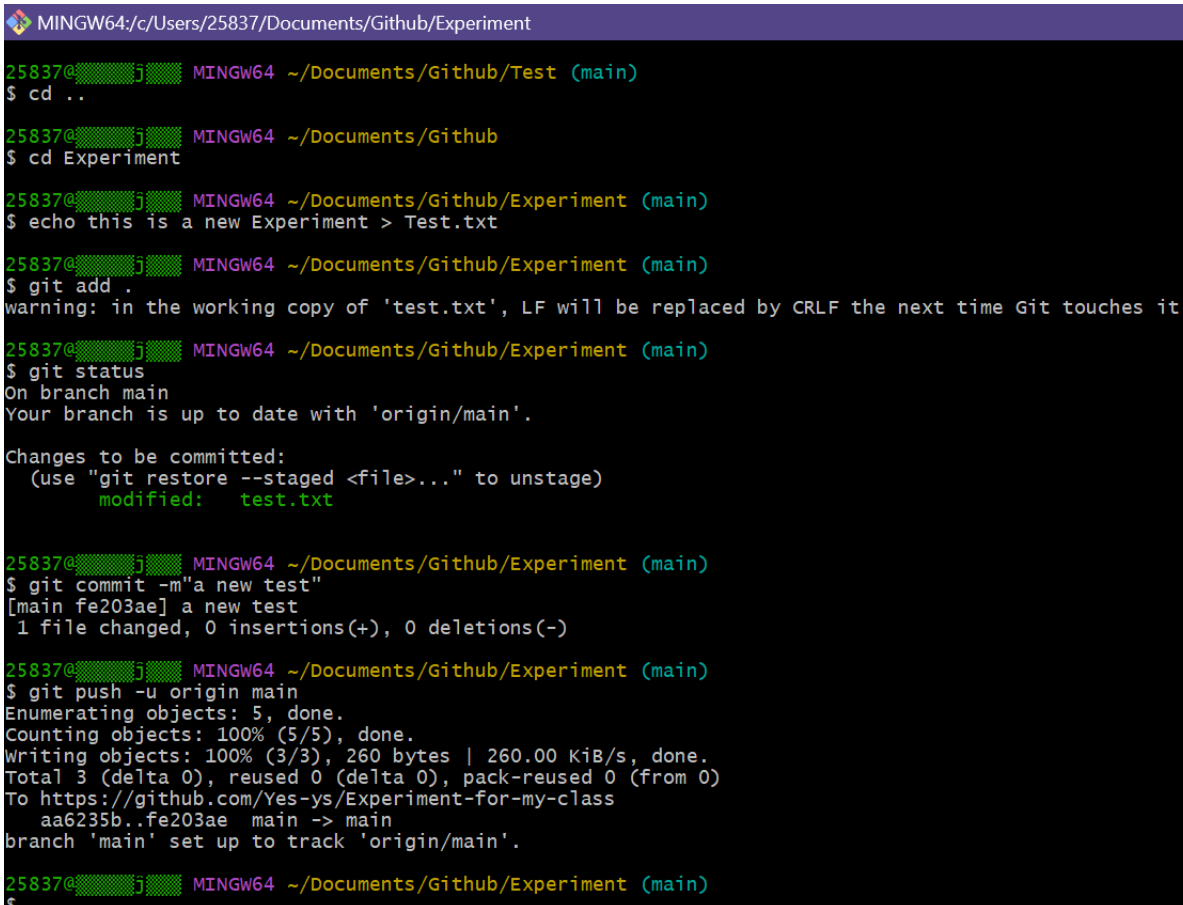
相关命令：

`git add .` 将整个工作区更新的内容存入暂存区

`git commit -m"提示信息"` 将暂存区内容提交

`git push -u origin main` 上传到Github

相关过程：



```
MINGW64/c/Users/25837/Documents/Github/Experiment

25837@MINGW64 ~/Documents/Github/Test (main)
$ cd ..

25837@MINGW64 ~/Documents/Github
$ cd Experiment

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ echo this is a new Experiment > Test.txt

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git add .
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git commit -m"a new test"
[main fe203ae] a new test
1 file changed, 0 insertions(+), 0 deletions(-)

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 260 bytes | 260.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Yes-ys/Experiment-for-my-class
   aa6235b..fe203ae  main -> main
branch 'main' set up to track 'origin/main'.

25837@MINGW64 ~/Documents/Github/Experiment (main)
$
```

图 6: Git Bash的操作

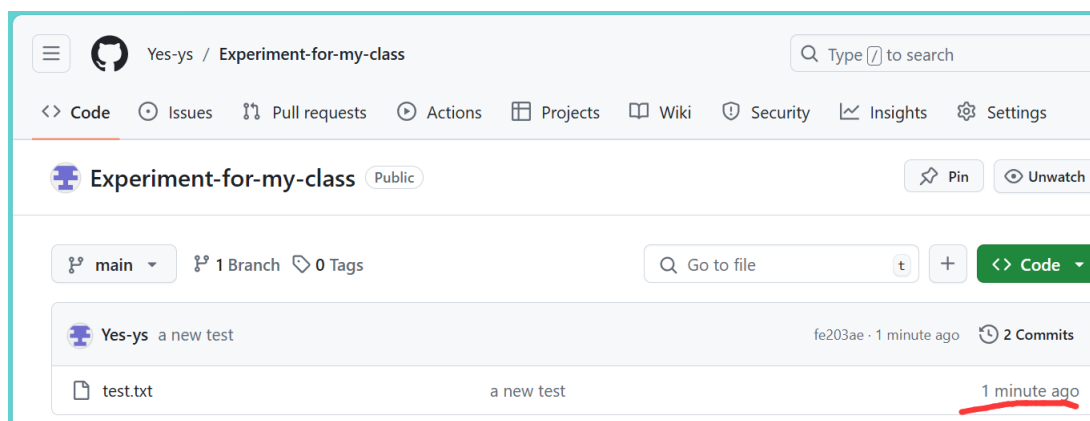


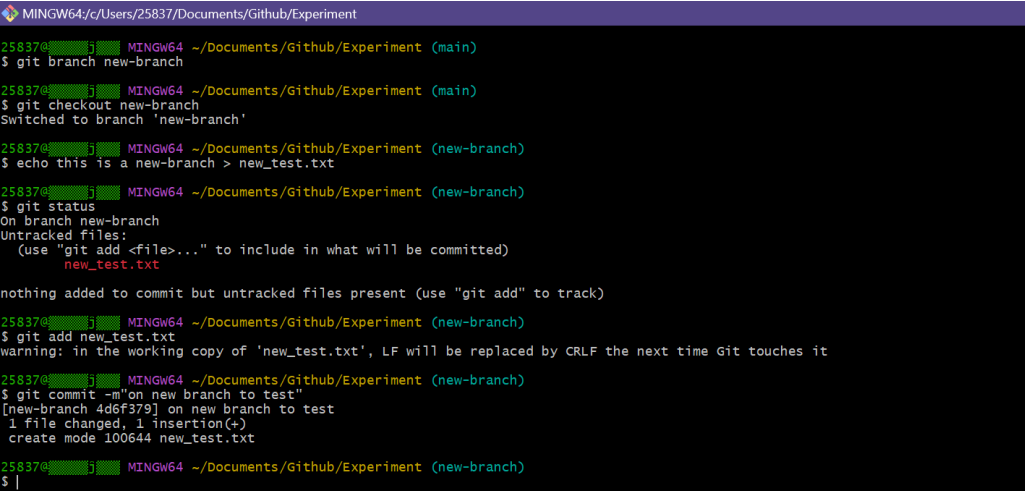
图 7: 操作后的结果

## 1.4 创建新分支并在新分支上进行操作

在Git Bash上进行操作，先创建一个新的分支，后以新分支在本地文件夹下新建文件，并提交。

命令如下：

```
git branch new-branch 创建新分支，名为new-branch
git checkout new-branch 将HEAD切换到new-branch
echo this is a new-branch > new_test.txt 新建txt文件
git add new_test.txt
git commit -m"on new branch to test" 提交
```



```
MINGW64/c/Users/25837/Documents/Github/Experiment
25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git branch new-branch
25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git checkout new-branch
Switched to branch 'new-branch'
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ echo this is a new-branch > new_test.txt
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ git status
On branch new-branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    new_test.txt

nothing added to commit but untracked files present (use "git add" to track)
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ git add new_test.txt
warning: in the working copy of 'new_test.txt', LF will be replaced by CRLF the next time Git touches it
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ git commit -m"on new branch to test"
[new-branch 4d6f379] on new branch to test
1 file changed, 1 insertion(+)
create mode 100644 new_test.txt
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ |
```

图 8: 操作后的结果



## 1.5 合并分支操作

将HEAD切换回main分支，再进行合并操作，相关命令如下：

`git switch main` 切换到main分支

`git merge new-branch` 合并分支new-branch

```
MINGW64/c/Users/25837/Documents/Github/Experiment
25837@MINGW64 ~/Documents/Github/Experiment (new-branch)
$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git merge new-branch
Updating fe203ae..4d6f379
Fast-forward
 new_test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new_test.txt

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Yes-ys/Experiment-for-my-class
   fe203ae..4d6f379  main -> main
branch 'main' set up to track 'origin/main'.

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ |
```

图 9: Git Bash上的操作

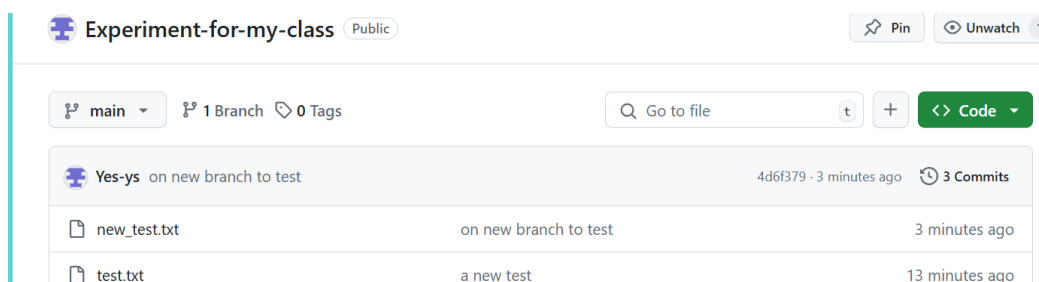


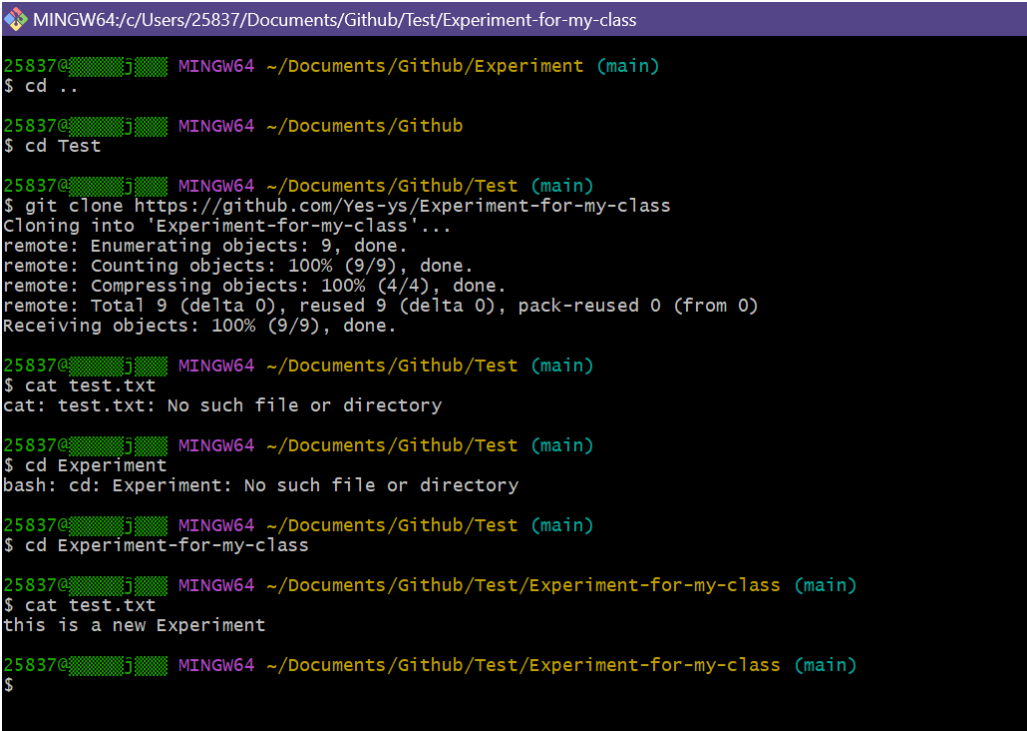
图 10: 操作后的结果（合并后执行了上传到Github中的操作）

## 1.6 从云端向本地的克隆操作

切换到最开始创建的本地仓库Test，克隆Github上的Repository，再使用cat查看克隆而来的txt文件

相关命令：

```
git clone Repository的网址
```



```
MINGW64/c/Users/25837/Documents/Github/Test/Experiment-for-my-class
25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ cd ..

25837@j MINGW64 ~/Documents/Github
$ cd Test

25837@j MINGW64 ~/Documents/Github/Test (main)
$ git clone https://github.com/Yes-ys/Experiment-for-my-class
Cloning into 'Experiment-for-my-class'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.

25837@j MINGW64 ~/Documents/Github/Test (main)
$ cat test.txt
cat: test.txt: No such file or directory

25837@j MINGW64 ~/Documents/Github/Test (main)
$ cd Experiment
bash: cd: Experiment: No such file or directory

25837@j MINGW64 ~/Documents/Github/Test (main)
$ cd Experiment-for-my-class

25837@j MINGW64 ~/Documents/Github/Test/Experiment-for-my-class (main)
$ cat test.txt
this is a new Experiment

25837@j MINGW64 ~/Documents/Github/Test/Experiment-for-my-class (main)
$
```

图 11: 操作及结果

---

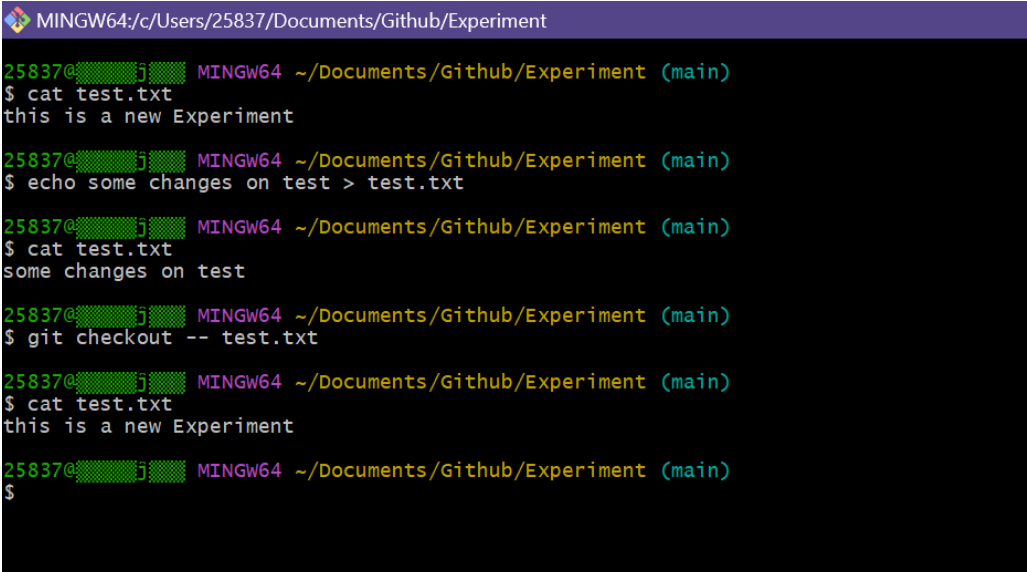
## 1.7 撤销对工作区文件的修改

切换回一般做实验用的本地仓库Experiment，对test文件进行修改后执行撤销操作，再检查其中的内容。

相关命令：

```
git checkout -- 文件名 撤销工作区文件的修改
```

```
git checkout -- test.txt 例如这样
```

A terminal window titled 'MINGW64:/c/Users/25837/Documents/Github/Experiment' showing a sequence of commands and their outputs. The user is in the 'main' branch. They first view 'test.txt' which contains 'this is a new Experiment'. Then they edit it with 'echo some changes on test > test.txt', resulting in 'some changes on test'. Finally, they run 'git checkout -- test.txt' to revert the changes, and the file content returns to 'this is a new Experiment'.

```
MINGW64:/c/Users/25837/Documents/Github/Experiment
25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$ cat test.txt
this is a new Experiment

25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$ echo some changes on test > test.txt

25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$ cat test.txt
some changes on test

25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$ git checkout -- test.txt

25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$ cat test.txt
this is a new Experiment

25837@j: MINGW64 ~/Documents/Github/Experiment (main)
$
```

图 12: 操作及结果

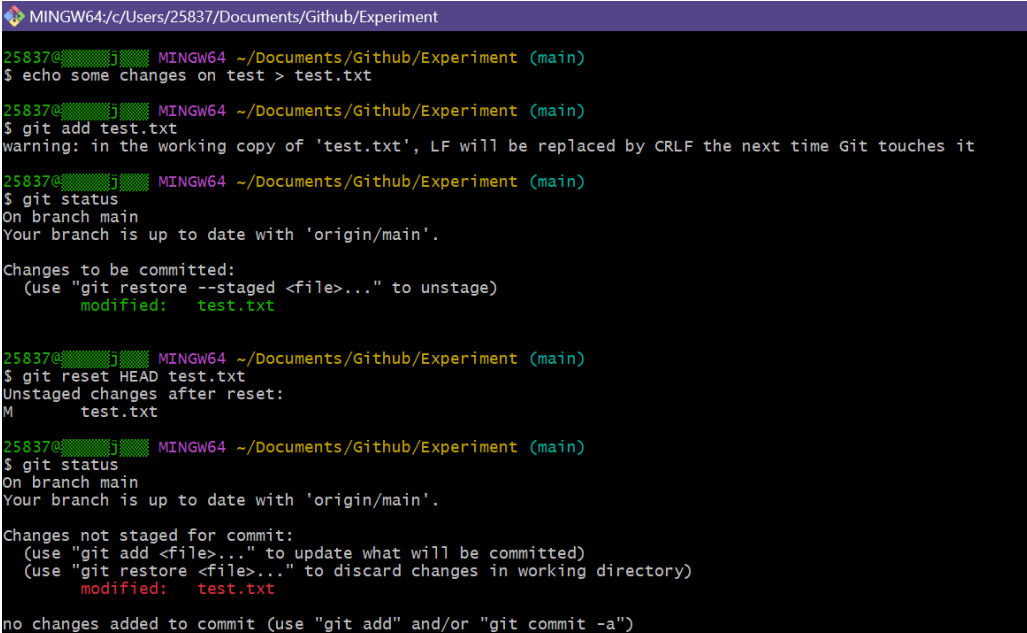
## 1.8 撤销对缓存区的修改

将新修改的工作区添加进入缓存区后，进行撤销操作。

相关命令：

`git reset HEAD` 加入缓存区的文件名 `撤销缓存区的修改` `git reset HEAD test.txt`

例如这样 `git status` 查看当前git的状态（显示实验的结果）

A terminal window titled 'MINGW64/c/Users/25837/Documents/Github/Experiment' showing a series of git commands and their outputs. The user first adds 'test.txt' to the index, then checks the status, which shows 'test.txt' as modified. Then, the user runs 'git reset HEAD test.txt', which unstages the changes. Finally, the user checks the status again, which shows 'test.txt' as modified but not staged for commit.

```
MINGW64/c/Users/25837/Documents/Github/Experiment
25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ echo some changes on test > test.txt

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   test.txt

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git reset HEAD test.txt
Unstaged changes after reset:
M   test.txt

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   test.txt

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

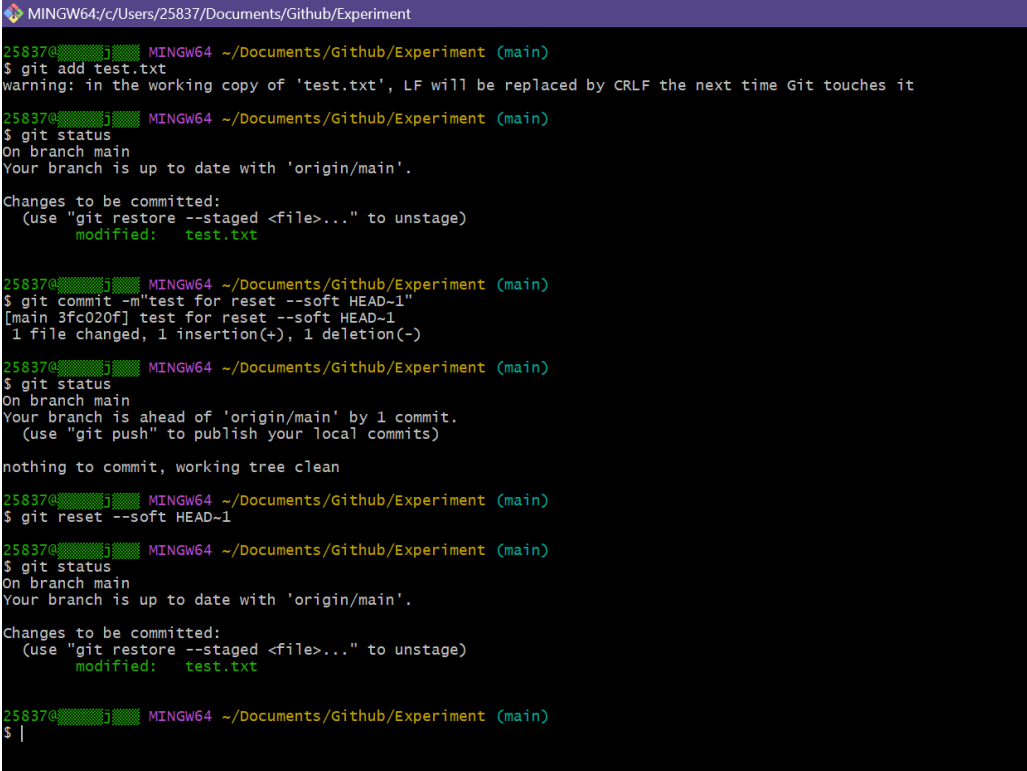
图 13: 操作及结果

## 1.9 撤销最新一次提交但保留暂存区记录

将最近的一次commit删除，但是保留这次commit之前的暂存区状态。

相关命令：

```
git reset -- soft HEAD 1 删除commit但保留暂存区
```



```
MINGW64/c:/Users/25837/Documents/Github/Experiment
25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it
25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   test.txt

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git commit -m"test for reset --soft HEAD-1"
[main 3fc020f] test for reset --soft HEAD-1
1 file changed, 1 insertion(+), 1 deletion(-)

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git reset --soft HEAD-1

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   test.txt

25837@MINGW64 ~/Documents/Github/Experiment (main)
$ |
```

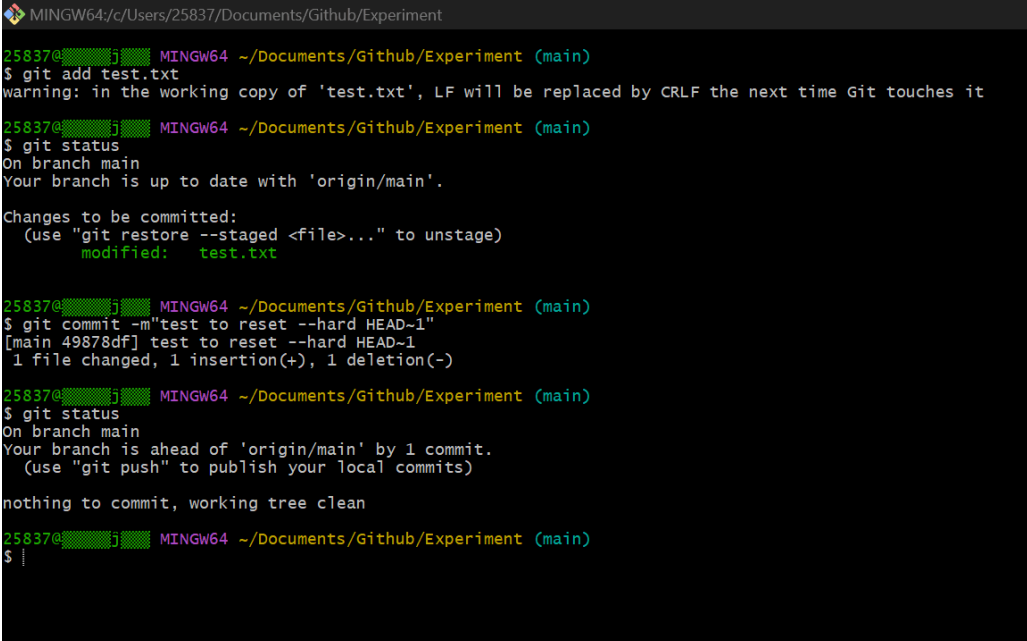
图 14: 操作及结果

## 1.10 撤销最新一次提交且删除缓存区记录

将最近一次commit删除，同时删除这次commit之前的暂存区状态。

相关命令：

```
git reset -- hard HEAD 1 删除commit同时情况暂存区
```



```
MINGW64/c:/Users/25837/Documents/Github/Experiment

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git commit -m"test to reset --hard HEAD~1"
[main 49878df] test to reset --hard HEAD~1
1 file changed, 1 insertion(+), 1 deletion(-)

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

25837@j MINGW64 ~/Documents/Github/Experiment (main)
$
```

图 15: 操作及结果

---

## 2 关于Latex的使用

### 2.1 文件的基本格式和标题的生成

每个latex文档，以命令 `\documentclass[a4paper, 12pt]{article}` 开始，`[]`是可选的，在里面可以设置你文档的纸张大小和主要字体大小。

紧随上个命令之后可以有一些先导命令，这些命令会影响整个文档。比如我使用命令 `\usepackage{xcolor}`，引入了可以让我自定义字体色彩的库，同时使用了 `\definecolor{lightgray}{RGB}{211,211,211}`，来在全局定义了我用来在展示相关命令和代码时使用的打字机字体后的灰色背景板。

最后，使用 `\begin{document}` 和 `\end{document}` 来包括文档所有的正文部分。

例如：

```
\begin{document}
Hello, LaTeX!
\end{document}
```

关于标题的生成，使用 `\title{这是一个标题}` 来开始标题生成，`\author{作者名}` 来显示作者名，`\date{相关日期}` 来显示日期，最后以 `\maketitle` 结尾。

例如本篇报告的标题部分代码如下：

```
\title{系统开发工具基础第一次实验报告}
\author{姓名：杨子程\ 学号：23020007145\ 专业：计算机科学与技术}
\date{\today}
\maketitle
```

### 2.2 目录的生成

目录生成的代码较为简单，即 `\tableofcontents`。

例如本篇报告的目录部分代码如下：

---

`\tableofcontents`

`\newpage` %用于开始新的一页，避免目录和正文在同一页

## 2.3 章节的生成

使用 `\chapter{本章的标题}`，来生成章；使用 `\section{本节的标题}`，来生成节；使用 `\subsection{子节的标题}`，来生成子节。往下还有 `\subsubsection{...}`、`\paragraph{...}`、`\subparagraph{...}` 等。

例如本篇报告生成章节的部分代码（在没加入正文之前，此subsection往下的部分）：

`\subsection{\color{red}彩色\colorbox{yellow}{\color{green}字体}\color{black}}`的使用}

`\subsection{自定义页码}`

`\subsection{列表（清单）的生成}`

`\subsection{表格的生成}`

`\subsection{图表的使用}`

`\subsection{插入各种数学公式}`

`\subsection{书写参考文献}`

## 2.4 彩色字体的使用

彩色字体的使用要先导入相关的包，例如color、xcolor包。常见的使用彩色字体的命令有 `\color{颜色代码}`正文部分，其中颜色代码常见的有red、green、blue等等...除此之外还有命令 `\clorbox{颜色代码}`正文部分，彩色背景板等...colorx中提供自定义颜色的命令 `\definecolor{lightgray}{RGB}{211,211,211}`例如这串代码自定义颜色为浅灰色。

本小节标题颜色的代码为：

`\color{red}彩色\colorbox{yellow}{\color{green}字体}\color{black}`的使用



---

## 2.5 自定义页码

我学习到的自定义页码可以使用包fancyhdr，其中提供了一些改变默认页码表示，自定义页码表示的方法。下面展示本篇报告中关于取消正下方中间显示页码，改为下方右侧显示页码的操作。

```
\usepackage{fancyhdr} % 引入 fancyhdr 宏包
\pagestyle{fancy}      % 使用 fancyhdr 提供的页面样式
\fancyhf               % 清除默认设置
\fancyfoot[R]{\thepage} % 将页码放在右下角
%以上操作都是针对整个文档进行的所以放在\begin{document}之前
```

## 2.6 列表（清单）的生成

使用 `\begin{enumerate}` 和 `\end{enumerate}` 作为一对，来开始与结束一个有序列表，对应将enumerate换为itemize使用无序列表，列表有些类似清单的感觉。在begin和end中间使用 `\item 项目名称` 来表示清单中的各项

下面展示一个我写的列表，安排我一天的任务

1. 学习数学建模
2. 完成系统开发基础工具的实验报告
  - Git部分
  - Latex部分
3. 完成计算机工程伦理的课程报告
4. 取快递
  - 数据线
  - 眼罩

以下是相关代码

---

```
\begin{enumerate}
\item 学习数学建模

\item 完成系统开发基础工具的实验报告
  \begin{itemize}
    \item Git部分

    \item Latex部分
  \end{itemize}

\item 完成计算机工程伦理的课程报告

\item 取快递
  \begin{itemize}
    \item 数据线

    \item 眼罩
  \end{itemize}
\end{enumerate}
```

## 2.7 表格的生成

使用 `\begin{tabular}{...}` 和 `\end{tabular}` 作为一对来生成表格。其中begin后的...部分由指定列相关内容的代码取代。涉及有 `l`, `r`, `c` 分别表示每列的对齐方式（左、右、中），还有 `|`，来表示分割列。在begin之后还可输入的有：`&`, `\\`, `\hline`, `\cline{...}`，分别表示分割列（与前面提到分割不同的是，前面是文档显示的分割，此处是代码逻辑的分割，因为&将会处于不同列的正文之间）；换行；分割行；分割指定行（...处为指定方法，常见的指定为 列1的序号-列2的序号，这样会添加一个分割行的线，从列1贯穿到列2）

例如，以下是我完成的参考资料上提供的实践题表格：

---

City	Year		
	2006	2007	2008
London	45789	46551	51298
Berlin	34549	32543	29870
Paris	49835	51009	51970

以下是上图的代码：

```
\begin{tabular}{l|ccc}
& Year & \\\
\cline{2-4}
City & 2006 & 2007 & 2008 & \\\
\hline
London & 45789 & 46551 & 51298 & \\\
Berlin & 34549 & 32543 & 29870 & \\\
Paris & 49835 & 51009 & 51970 & \\
\end{tabular}
```

## 2.8 图表的使用

图表插入要用到包graphicx，例如在我本篇报告的第一个实验插入的第一张图片，使用了如下代码：

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{im1}
\caption{本地文件夹}
\label{image-myimage}
\end{figure}
```

其中涉及到`\centering`，表示插入图片的位置，centering表示居中。对于`\includegraphics[]{}{}`，[]是可选的，用于设置图片的大小尺寸、大括号中包括了图片的地址，当图片与latex文档在同一文件夹下时只需要文件名即可`\caption{}{}`用于设置图片名，大括号中书写。最后使用label添加了一

---

个标签，这是可选的。以上提到的都要插入 `\begin{figure}` 和 `\end{figure}` 对之间，这对代码表示要开始插入图片。

## 2.9 插入各种数学公式

数学公式的插入使用 `$ $` 或 `$$ $$` 进行标记。常见的命令有，表示上下标的 `^` 和 `_`；表示分数的 `\frac{分子}{分母}`；表示根号的 `\sqrt{...}`；表示求和的 `\sum`；表示积分的 `\int` 等等的数学公式

下面是我就参考资料的实践题进行的实现：

$$e = mc^2 \quad (1)$$

$$\pi = \frac{c}{d} \quad (2)$$

$$\frac{d}{dx} e^x = e^x \quad (3)$$

$$\frac{d}{dx} \int_0^\infty f(s) ds = f(x) \quad (4)$$

$$f(x) = \sum_i 0^\infty \frac{f^{(i)}(0)}{i!} x^i \quad (5)$$

$$x = \sqrt{\frac{x_i}{z} y} \quad (6)$$

以下是上面公式的代码：

```
\begin{align}
e &= mc^2 \\
\pi &= \frac{c}{d} \\
\frac{d}{dx} e^x &= e^x \\
\frac{d}{dx} \int_0^\infty f(s) ds &= f(x) \\
f(x) &= \sum_i 0^\infty \frac{f^{(i)}(0)}{i!} x^i \\
x &= \sqrt{\frac{x_i}{z} y}
\end{align}
```

---

## 2.10 书写参考文献

我在按照参考网站引用参考文献的时候发生错误，经上网查阅发现是因为我的这份latex文档名字中包含了中文字符（非ASCII字符），导致一些辅助文件例如.aux文件也含有这些字符，从而导致了编译失败。下面是我新建了一个latex文档，书写的有关参考文献的内容：

### References

- [1] Albert Einstein. On the electrodynamics of moving bodies. *Annalen der Physik*, 322(10):891–921, 1905.

图 16: 如上

相关代码：

```
\bibliographystyle{plain}  
\bibliography{MyReference}
```

第一行说明了引用文献时的排版样式，第二个MyReference是当前目录下的bib文件名，书写参考文献用的文件。注意，在有参考文献的时候，要先执行BibTex编译，再使用正常的pdfLaTex编译，这样才会得到正确的结果。