



课程作业报告

课程名称：系统工具开发基础

学生姓名：张家玮

学号：23020007157

日期：2024 年 8 月 30 日



目录

1	LaTeX的学习和使用	3
1.1	设计一个简单的模板	3
1.2	正文的相关设置	5
1.2.1	标题和章节管理	5
1.2.2	正文文本格式	6
1.2.3	行距和段距控制	6
1.2.4	脚注和颜色	7
1.3	插入数学公式	7
1.4	插入图片	8
1.5	列表	9
1.6	表格	10
1.7	插入代码	11
2	Git的学习和使用	12
2.1	克隆仓库	12
2.1.1	将版本历史可视化并进行探索	13
2.1.2	是谁最后修改了 README.md 文件?	14
2.1.3	最后一次修改时的提交信息是什么?	14
2.2	向仓库中添加文件提供信息, 然后从历史中删除	15
2.3	关于git stash	16
2.4	配置文件中创建别名	16
2.5	配置.gitignore文件	17
2.6	Fork and PR	18
3	学习心得	20
3.1	感悟	20
3.2	学习中遇到的困难	20
3.2.1	LaTeX	20
3.2.2	Git	20



Chapter 1

LaTeX的学习和使用

1.1 设计一个简单的模板

这份报告使用的模板即为本次练习设计的模板，由封面，正文各章节的内容和最后的参考文献标注组成。

在实际的LaTeX代码实现中主要分为以下几个块组成

- 定义文档类型
- 设置中文支持和相关包导入
- 页眉页脚设置
- 文档的主题内容

代码示例如下

```
\documentclass[a4paper,12pt]{report} % 定义文档类型

% 中文支持及其他常用包
\usepackage[UTF8]{ctex} % 中文支持
\usepackage{geometry} % 页面边距设置
\geometry{left=2.5cm, right=2.5cm, top=2.5cm, bottom=2.5cm}
\usepackage{graphicx} % 插入图片
\usepackage{amsmath} % 数学公式
\usepackage{hyperref} % 超链接
\usepackage{fancyhdr} % 页眉页脚
\usepackage{titlesec} % 控制标题格式
\usepackage{listings} % 插入代码
\usepackage{setspace} % 行距
```



```
% 页眉页脚设置
\pagestyle{fancy}
\fancyhf{}
\renewcommand{\headrulewidth}{1pt} % 页眉下方线条宽度
\renewcommand{\footrulewidth}{1pt} % 页脚下方线条宽度
\fancyhead[L]{\includegraphics[width=1.5cm]{logo.png}} % 在页眉左侧插入
学校 logo
\fancyhead[C]{课程作业报告} % 在页眉中间显示
\fancyfoot[C]{\thepage} % 在页脚中间显示页码

% 文档开始
\begin{document}

% 封面页
\begin{titlepage}
  \centering
  \vspace*{2cm}
  \includegraphics[width=4cm]{ouc.png}\\
  \vspace{1cm}
  {\Huge \textbf{课程作业报告}}\\[1.5cm]
  {\Large 课程名称：系统工具开发基础}\\[1cm]
  {\Large 学生姓名：张家玮}\\[1cm]
  {\Large 学号：23020007157}\\[1cm]
  {\Large 日期：\today}
  \vfill
\end{titlepage}

% 目录页
\tableofcontents

\newpage

% 正文部分
\chapter{LaTeX的学习和使用}
\section{设计一个简单的模板}

% 文档结束
\end{document}
```

按照如上的代码成功设计出一个建议的报告模板。



1.2 正文的相关设置

关于LaTeX的正文部分所有都包括在begin...end命令中间的区域，最终所有的可见内容也都在此区域中添加，包括文字。下面我将会通过一个实例来展示相应的知识。

1.2.1 标题和章节管理

如下列代码所示

```
\chapter{标题和章节管理}
```

```
\section{一级标题示例}
```

这是一级标题的内容。接下来是一个二级标题。

```
\subsection{二级标题示例}
```

这是二级标题的内容。以下展示文本的基本格式化。

```
\subsubsection{三级标题示例}
```

这是三级标题的内容。

```
\paragraph{四级标题示例}
```

这是四级标题的内容。

得到的效果如图所示

Chapter 1

标题和章节管理

1.1 一级标题示例

这是一级标题的内容。接下来是一个二级标题。

1.1.1 二级标题示例

这是二级标题的内容。以下展示文本的基本格式化。

三级标题示例

这是三级标题的内容。

四级标题示例 这是四级标题的内容。



1.2.2 正文文本格式

如下列代码所示

```
\chapter{正文文本格式}
```

```
\section{文本样式}
```

这是一个普通文本。\\

这是 \textbf{加粗} 的文本。\\

这是 \textit{斜体} 的文本。\\

这是 \underline{带下划线} 的文本。\\

这是 \textcolor{blue}{蓝色} 的文本。

得到的效果如图所示

正文文本格式

2.1 文本样式

这是一个普通文本。

这是 **加粗** 的文本。

这是 *斜体* 的文本。

这是 带下划线 的文本。

这是 蓝色 的文本。

1.2.3 行距和段距控制

关于行距和段距控制有如下指令可以使用，这里不再展示具体效果

```
\onehalfspacing
```

这是1.5倍行距。\\

```
\doublespacing
```



这是双倍行距。`\`

`\singlespacing`

这是恢复为单倍行距。

`\setlength{\parskip}{1em}` % 设置段间距为1em,

`\`设置以后这段文字之前和之后都会有段落距离。

1.2.4 脚注和颜色

脚注和颜色的相关命令如下例所示

这是一个带脚注的文本`\footnote{这是脚注的内容}`,

还可以改变文本颜色, 比如 `\textcolor{red}{红色文本}`。

1.3 插入数学公式

LaTeX中的插入公式操作主要分为以下两种:

- **行内公式:** 使用美元符号 `...` 将数学公式包裹起来, 公式将与普通文本在同一行内显示。
- **单独公式:**

使用`\[... \]`或 `equation` 环境将公式单独显示在一行。

该环境也会自动给公式标号, 如果不想标号的话就要在环境使用的标注符后面加上`*`号。

下面是我写的一些相应的公式的实例



这是一个简单的行内公式: $a + b = b + a$,

这是一个单独显示的公式:

$$a + b = b + a$$

这是一个带编号的公式:

$$\int_0^{\infty} e^x dx = 1 \quad (1)$$

如公式 1 所示, 这是一个不定积分。

这是一个分数公式:

$$\frac{a + b}{c + d}$$

上下标的使用示例:

$$x_1, \quad y^2, \quad z_i^n$$

这是平方根符号和 n 次方根符号的示例:

$$\sqrt{x}, \quad \sqrt[n]{y}$$

积分和求和的示例:

$$\int_a^b f(x) dx, \quad \sum_{i=1}^n i^2$$

使用 'align' 环境对齐多行公式:

$$f(x) = x^2 + 2x + 1 \quad (2)$$

$$g(x) = x^2 - 2x + 1 \quad (3)$$

一些常见的希腊字母: $\alpha, \beta, \gamma, \delta, \pi, \Sigma, \Omega$

1.4 插入图片

在LaTeX插入图片的操作中需要先导入宏包'graphicx' 插入图片的格式一般如下

```
\begin{figure}[h] % [h] 表示在当前位置插入图片
```

```
\centering % 居中
```

```
\includegraphics[width=\textwidth]{path/to/image.png} % 图片路径和大小
```

```
\caption{这是图片的说明文字} % 图片说明
```

```
\label{fig:example} % 图片标签, 用于引用
```

```
\end{figure}
```

在这里我们插入我校校徽来演示



图 1.1: 这是中国海洋大学的校徽

1.5 列表

关于列表主要分为无序列表，有序列表，描述性列表。
如下实例

```
\section{列表}
```

```
\subsection{无序列表}
```

```
\begin{itemize}
```

```
\item 项目1
```

```
\item 项目2
```

```
\item 项目3
```

```
\end{itemize}
```

```
\subsection{有序列表}
```

```
\begin{enumerate}
```

```
\item 第一点
```

```
\item 第二点
```

```
\item 第三点
```

```
\end{enumerate}
```

```
\subsection{描述性列表}
```

```
\begin{description}
```



```
\item[术语A] 这是术语A的定义。
\item[术语B] 这是术语B的定义。
\end{description}
```

得到的效果如图所示

2.2 列表

2.2.1 无序列表

- 项目 1
- 项目 2
- 项目 3

2.2.2 有序列表

1. 第一点
2. 第二点
3. 第三点

2.2.3 描述性列表

术语 A 这是术语 A 的定义。

1.6 表格

在LaTeX中创建表格，通常使用'tabular'环境。可以控制表格的对齐方式、边框样式以及单元格内容。如下例展示

```
\begin{tabular}{|l|c|r|}
\hline
姓名 & 课程 & 成绩 \\ \hline
张三 & 数学 & 90 \\
李四 & 英语 & 85 \\
王五 & 物理 & 92 \\
\hline
\end{tabular}
```

得到的效果如图所示



姓名	课程	成绩
张三	数学	90
李四	英语	85
王五	物理	92

1.7 插入代码

在LaTeX中插入代码通常使用'listings'宏包。然后配置自定义代码的格式，然后在文档中使用'listings'环境插入代码我们这里用C++代码来展示展示效果如下：

C++

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     //      "Hello, World!"
6     cout << "Hello, World!" << endl;
7     return 0;
8 }
```

Listing 1: C++



Chapter 2

Git的学习和使用

2.1 克隆仓库

- 安装Git
- 获取仓库的克隆链接
- 打开终端，在windows系统中我们使用”Git Bash”打开终端
- 使用’git clone’命令克隆仓库

```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice
$ git clone git@github.com:missing-semester-cn/missing-semester-cn.github.io.git
Cloning into 'missing-semester-cn.github.io'...
Enter passphrase for key '/c:/Users/23724/.ssh/id_rsa':
remote: Enumerating objects: 3194, done.
remote: Counting objects: 100% (3194/3194), done.
remote: Compressing objects: 100% (1126/1126), done.
remote: Total 3194 (delta 2040), reused 2735 (delta 2033), pack-reused 0 (from 0)
Receiving objects: 100% (3194/3194), 15.44 MiB | 186.00 KiB/s, done.
Resolving deltas: 100% (2040/2040), done.
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice
$ |
```

我们切换到克隆的仓库所在的目录查看文件列表



```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/missing-seme...
remote: Enumerating objects: 3194, done.
remote: Counting objects: 100% (3194/3194), done.
remote: Compressing objects: 100% (1126/1126), done.
remote: Total 3194 (delta 2040), reused 2735 (delta 2033), pack-reused 0 (from 0)
Receiving objects: 100% (3194/3194), 15.44 MiB | 186.00 KiB/s, done.
Resolving deltas: 100% (2040/2040), done.

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice
$ cd /c:/Users/23724/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.github.io

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.github.io (master)
$ ls
404.html      _2019/        about.md      index.md
CNAME         _2020/        apple-touch-icon.png  lectures.html
Gemfile       _config.yml   favicon-16x16.png    license.md
Gemfile.lock  _includes/    favicon-32x32.png    robots.txt
README.md     _layouts/     favicon.ico        static/

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.github.io (master)
$ |
```

克隆成功

2.1.1 将版本历史可视化并进行探索

我们使用 'git log -all -graph -decorate' 命令来对历史进行可视化探索

- '-all': 显示所有分支的提交历史记录, 包括远程分支和本地分支的所有提交。
- '-graph': 以图形方式显示提交历史的分支结构。这使得查看分支和合并的情况更加直观。
- '-decorate': 显示每个提交的附加信息。

得到的历史记录如下

```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/missing-seme...
* commit af054fa1aea2f2599e4474d96b63f73dd9bd145f (HEAD -> master, origin/master, origin/HEAD)
Merge: dd3f3dd 9baa48c
Author: Lingfeng_Ai <hanxiaomax@gmail.com>
Date: Fri Aug 16 06:54:16 2024 +0800

    Merge pull request #172 from pspdada/master

    Thank you so much

* commit 9baa48c778012164179e4e60725418941f41743b
Author: psp_dada <1824427006@qq.com>
Date: Thu Aug 15 02:07:36 2024 +0800

    remove irrelevant text

* commit f5df7de89dc7712483665cc6fe8a787aafbef9bf
Author: psp_dada <1824427006@qq.com>
Date: Thu Aug 15 01:46:12 2024 +0800

    fix wrong index

* commit ef9a2f75409ff7746c03f6233066e3d2c634cd12
```

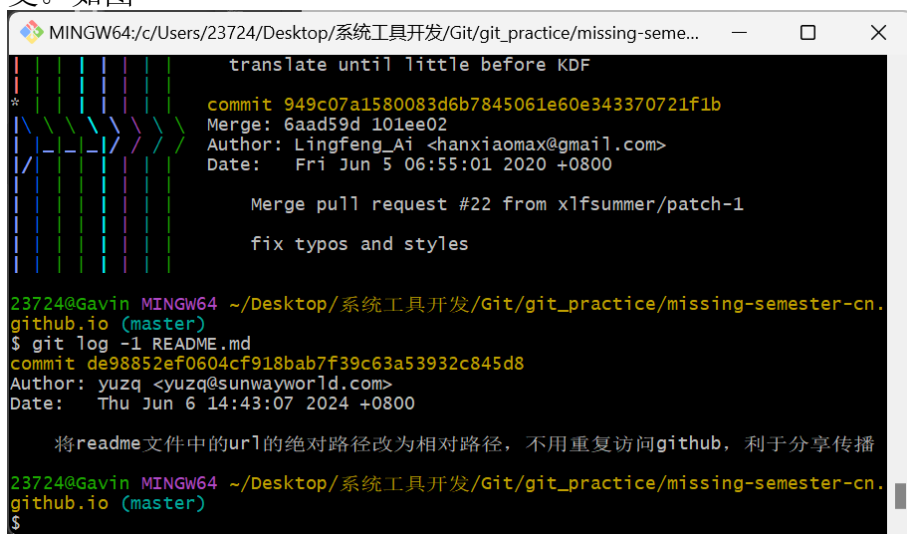


2.1.2 是谁最后修改了 README.md 文件？

在这里给出一些方法来查看修改

- 'git log -1 -- README.md':查看最后一次修改。
- 'git blame README.md':查看每一行的最后修改者。
- 'git log --follow -p -- README.md':查看完整的文件修改历史。

我们这里使用'git log -1 -- README.md'指令，这里的'-1'表示限制输出为最近的一次提交。如图



```
MINGW64~/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/missing-semester-cn...
translate until little before KDF
*
commit 949c07a1580083d6b7845061e60e343370721f1b
Merge: 6aad59d 101ee02
Author: Lingfeng_Ai <hanxiaomax@gmail.com>
Date: Fri Jun 5 06:55:01 2020 +0800

Merge pull request #22 from xlfsummer/patch-1

fix typos and styles

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git log -1 README.md
commit de98852ef0604cf918bab7f39c63a53932c845d8
Author: yuzq <yuzq@sunwayworld.com>
Date: Thu Jun 6 14:43:07 2024 +0800

将readme文件中的url的绝对路径改为相对路径，不用重复访问github，利于分享传播

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$
```

2.1.3 最后一次修改时的提交信息是什么？

在这里我们使用git blame config.yml grep collections这个指令用于查找 config.yml 文件中包含字符串 collections 的行，并显示这些行的最后一次修改者及其相关信息。

得到的最后一次的修改信息我们用'git show'指令来查看

如图所示



```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/missing-seme...
github.io (master)
$ git blame _config.yml | grep collections
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 18) collections:

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git show a88b4eac
commit a88b4eac326483e29bdac5ee0a39b180948ae7fc
Author: Anish Athalye <me@anishathalye.com>
Date: Fri Jan 17 15:26:30 2020 -0500

    Redo lectures as a collection

diff --git a/2020/index.html b/2020/index.html
deleted file mode 100644
index 153ddc8..0000000
--- a/2020/index.html
+++ /dev/null
@@ -1,5 +0,0 @@
-----
-layout: redirect
-redirect: /
-title: Missing Semester 2020
-----
```

2.2 向仓库中添加文件提供信息，然后从历史中删除

我们首先提交一个secret.txt文件并保存好提交信息，我们先在终端中导航到我们的Git仓库中然后使用echo指令创建文件，然后通过git status指令查看当前仓库状态，可以看到secret.txt文件发生了修改再通过git add 和 git commit 指令提交更改,并用git push提交到远程仓库成功提交

```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/Git_practice
warning: in the working copy of 'secret.txt', LF will be replaced by CRLF the next time Git touches it

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git commit -m "error submission"
[main 5450214] error submission
1 file changed, 1 insertion(+)
create mode 100644 secret.txt

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git push origin main
Enter passphrase for key '/c:/Users/23724/.ssh/id_rsa':
Enter passphrase for key '/c:/Users/23724/.ssh/id_rsa':
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), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Gavinnn/Git_practice.git
34e0347..5450214 main -> main

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$
```

然后我们进行删除操作使用BFG Repo-Cleaner工具进行操作首先从其官网上下载对应的bfg.jar放于我们的目录下便于我们进行操作。使用:

```
java -jar bfg.jar --delete-secret.txt Git_practice.git
git reflog expire --expire=now --all && git gc --prune=now --aggressive
```

指令来删除我们之前提交的错误文件。再使用git中的指令来清理历史记录，最后强制推送到远程仓库就可以完成所有的操作。



2.3 关于git stash

我们将自己在GitHub上建立的用于练习git操作指令的远程仓库克隆到本地然后我们修改一下本来在其中的secret.txt文件。然后我们执行如图下面的指令来探索下git stash的作用使用nano命令进入我们要修改的文件，我们做一些小的修改然后提交，然后我们使用git stash可以看到它会将我们之前未提交的修改保存到一个临时的区域，并将工作目录恢复到干净的状态。我们查看提交记录会发现这次提交并没有出现使用 git stash pop指令会恢复最近的 git stash，并将其应用到当前工作目录，同时将该存档从stash 列表中移除。

```
MINGW64:/c:/Users/23724/Desktop/系统工具开发/Git/git_practice/Git_practice

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ nano secret.txt

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (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:   secret.txt

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

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git add secret.txt

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git stash
Saved working directory and index state WIP on main: 5f35c61 Add secret.txt

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git log --all --oneline
f2df5b9 (refs/stash) WIP on main: 5f35c61 Add secret.txt
8505d68 index on main: 5f35c61 Add secret.txt
5f35c61 (HEAD -> main, origin/main, origin/HEAD) Add secret.txt
b409220 Delete secret.txt
5450214 error submission
34e0347 Create README.md

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git stash pop
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:   secret.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (f2df5b9d8c84bb1ad18c7519fb024ec6ad4453ad)

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$
```

2.4 配置文件中创建别名

Git提供给了我们一个配置文件我们可以在该文件中创建别名可以让命令达到你所希望的效果,我们先通过git指令查找到自己的配置文件的路径，然后使用nano命令进入，创建别名。编辑完成以后我们再使用cat指令查看一下我们的.gitconfig 文件，如图



```
MINGW64/c/Users/23724

23724@Gavin MINGW64 ~
$ git config --list --show-origin
file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true
file:C:/Program Files/Git/etc/gitconfig http.sslbackend=openssl
file:C:/Program Files/Git/etc/gitconfig http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true
file:C:/Program Files/Git/etc/gitconfig core.fscache=true
file:C:/Program Files/Git/etc/gitconfig core.symlinks=false
file:C:/Program Files/Git/etc/gitconfig pull.rebase=false
file:C:/Program Files/Git/etc/gitconfig credential.helper=manager
file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com.usehttp
path=true
file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/23724/.gitconfig user.name=Gavlnnn
file:C:/Users/23724/.gitconfig user.email=2372460544@qq.com
file:C:/Users/23724/.gitconfig credential.helper=store
file:C:/Users/23724/.gitconfig http.sslcainfo=C:/Program Files/Git/mingw64/ssl/
certs/ca-bundle.crt
file:C:/Users/23724/.gitconfig http.sslverify=false

23724@Gavin MINGW64 ~
$ nano C:/Users/23724/.gitconfig

23724@Gavin MINGW64 ~
$ cat ~/.gitconfig
[user]
  name = Gavlnnn
  email = 2372460544@qq.com
[credential]
  helper = store
[http]
  sslCAinfo = C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
  sslverify = false
[filter "lfs"]
  required = true
  clean = git-lfs clean -- %f
  smudge = git-lfs smudge -- %f
  process = git-lfs filter-process
[alias]
  graph = log --all --graph --decorate --oneline
```

这样在运行 `git graph` 时，我们可以得到 `git log --all --graph --decorate --oneline` 的输出结果。

2.5 配置.gitignore文件

这里我们配置一个全局的忽略规则，来配置全局gitignore文件来自动忽略系统或者编辑器的临时文件，写入*.log用于忽略所有 .log类型的文件。

```
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ touch ~/.gitignore_global

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git config --global core.excludesfile ~/.gitignore_global

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ nano ~/.gitignore_global
```

如何我们写入一个用于测试的.log文件再尝试提交操作



```
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ echo "This is a test file" > test.log

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (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:   secret.txt

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

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/Git_practice (main)
$ git add test.log
The following paths are ignored by one of your .gitignore files:
test.log
hint: Use -f if you really want to add them.
hint: Disable this message with "git config advice.addIgnoredFile false"
```

可以看到我们的忽略规则已经生效。

2.6 Fork and PR

我们先在自己的GitHub账户上面Fork本课程的仓库，然后再将Fork好的仓库克隆到本地，进入仓库目录然后我们修改文件，提交修改并推到远程仓库中

```
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git status
On branch master
Your branch is up to date with 'origin/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:   about.md

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

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git add about.md

23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git commit -m "just for study"
[master e396af9] just for study
1 file changed, 1 insertion(+), 1 deletion(-)

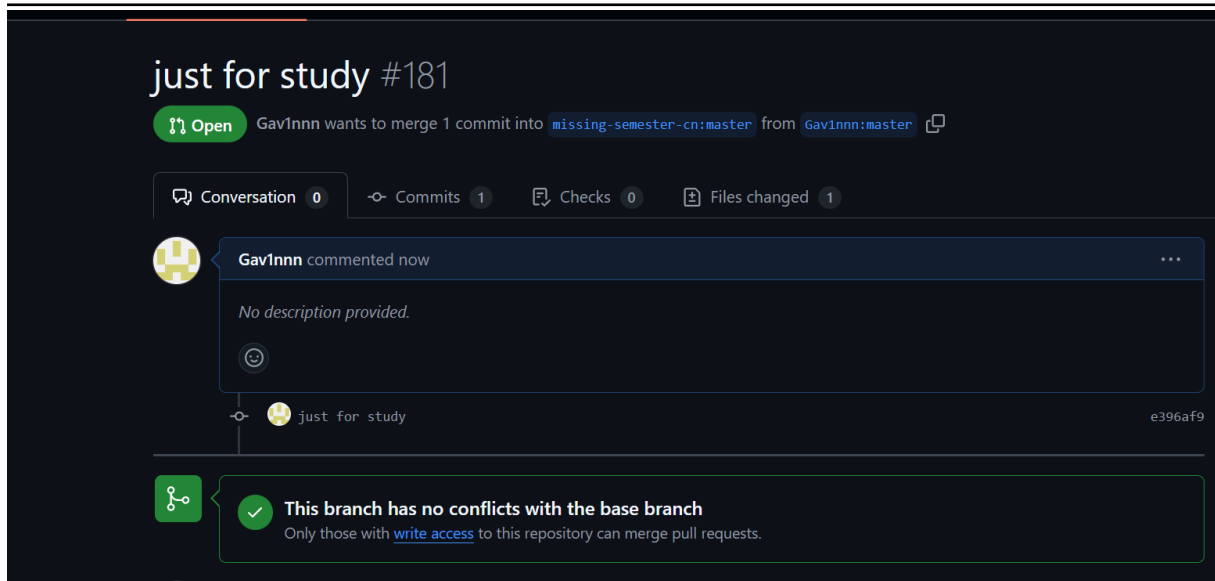
23724@Gavin MINGW64 ~/Desktop/系统工具开发/Git/git_practice/missing-semester-cn.
github.io (master)
$ git push origin master
Enter passphrase for key '/c/Users/23724/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:GavInnn/missing-semester-cn.github.io.git
   af054fa..e396af9  master -> master
```

然后我们登入GitHub发起拉取请求，填写拉去请求信息，最后发起请求。

如图，请求已经发送



课程作业报告



可看到拉取请求已经成功发送。



Chapter 3

学习心得

3.1 感悟

本次小学期所修的系统工具开发基础正如课程引言中讲的，这门课程是一门旨在帮助我们十分高效的处理自己的工作。本周学习使用了LaTeX和Git。LaTeX用来编辑排版自己的报告，论文等。这是一个很强大的排版工具，不仅能将文章排版得非常有序，同时在我们使用这种编辑模式，也能让我更加专注在内容上，而不是浪费精力在排版上。Git也是一个强大的分布式管理代码的工具，我们可以通过Git来更好地管理自己的代码，更高效得完成修改，调试等工作。学习新东西的开始总是痛苦的，但是当我们成功学习入门之后，这些工具带给了我们很大的便利，能够让我们十分高效得完成自己的工作。

3.2 学习中遇到的困难

3.2.1 LaTeX

在LaTeX中遇到的最大困难就是布置相应的环境，个人使用的是Tex Live然后配置VScode后，通过vscode来进行相关的.tex文件编辑。在软件都下载好之后，还需要设置vscode的环境，主要通过查询维基百科和谷歌来完成。后面LaTeX中遇到的问题也大多是各种指令需要其对应的宏包，我通过搜索引擎等解决了问题。

3.2.2 Git

在关于版本控制的学习中，困难主要在于设置自己GitHub中相应的SSH密钥的设置，以及清理工具BFG的使用，还有一些相关的操作指令，主要通过看我们课程所发的导航以及其中所含的相应文章，还有维基百科上的一些内容，以及各种搜索引擎解决。



Chapter 4

参考资料

- Missing Semester CN
- Google
- Wikipedia
- Missing Semester GitHub Repository
- BFG Repo-Cleaner

如果想要查看相关实例练习的源代码可以查询我的GitHub仓库
GitHub Repository