

系统开发工具基础 | Git & Latex

梁子毅 22090001041

Instructor: 周小伟, 范浩, 2024 Summer | **截止时间:** Aug 28, 2024

一、Git

1. 课后练习

问题 1：克隆本课程网站的仓库

可以使用 `git clone https://github.com/missing-semester-cn/missing-semester-cn.github.io.git` 来克隆网站的仓库，运行结果如下图 1

```
Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester
$ git clone https://github.com/missing-semester-cn/missing-semester-cn.github.io.g
it
Cloning into 'missing-semester-cn.github.io'...
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 | 2.51 MiB/s, done.
Resolving deltas: 100% (2040/2040), done.
```

图 1 `git clone`

问题 2：将版本历史可视化并进行探索

可以使用 `git log --all --graph --decorate` 进行可视化，效果如图 2

[illegible]

图 2 `git log --all --graph --decorate`

问题 3: 探索谁最后修改了 **README.md** 文件? (提示: 使用 `git log` 命令并添加合适的参数)

有两种方式：

1. 直接用 `git log` 打开，会产生一个类似 vim 的编辑器，使用 `/README.md` 可以找到这个文件
2. 或者使用 `git log -1 README.md` 可以读取含有 README.md 的第一行如图图 3

```

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.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

```

图3 探索谁最后修改了 README.md 文件

问题 4：最后一次修改 `_config.yml` 文件中 `collections:` 行时的提交信息是什么？（提示：使用 `git blame` 和 `git show`）

1. 可以使用 `git blame _config.yml | grep collections` 来显示
2. 也可以使用 `git show --pretty=format:"%s" a88b4eac | head -1`
3. 二者运行如下图 4

```

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.io (master)
$ git blame _config.yml | grep collections
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 18) collections:

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.io (master)
$ git show --pretty=format:"%s" a88b4eac | head -1
Redo lectures as a collection

```

图4 最后一次修改 `_config.yml` 文件中 `collections:` 行时的提交信息

问题 5：从 GitHub 上克隆某个仓库，修改一些文件。当您使用 `git stash` 会发生什么？通过 `git stash pop` 命令来撤销 `git stash` 操作，什么时候会用到这一技巧？

1. 使用 `git stash` 的时候，会将当前工作区清空，git 会将刚刚工作区的文件存起来。
2. 当你完成其他分支的编写（比方说 bug 处理）再用 `git stash pop` 取出你之前隐藏的文件（比方说正在开发新功能的文件）

问题 6：当您执行 `git log --all --oneline` 时会显示什么？

运行结果如下图 5，因为输出太长，使用 `tail -n 10` 和管道来输出最后的 10 行内容

```

37020ca Bug fix in command-line.md
497666a Merge pull request #151 from 2090741942/master
1de68eb please ignore

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.io (master)
$ git log --all --oneline | tail -n 10
bd94bf5 Fix parsing
e7dd5e1 First draft of shell lecture
a524147 Add under-construction page
86f74ec Add content for VMS
eb14a91 Add some intro content
db290a4 Add nicer message
ccd5a83 Acknowledge sponsors
ffa24fb Add schedule and empty pages for topics
b2e3a22 Add basic course info
112ddbd Initial commit

```

图5 `git log --all --oneline` 的运行结果

问题 7：与其他的命令行工具一样，Git 也提供了一个名为 `/.gitconfig` 配置文件 (或 dotfile)。请在 `/.gitconfig` 中创建一个别名，使您在运行 `git graph` 时，您可以得到

`git log --all --graph --decorate --oneline` 的输出结果；

1. 用 `vim ~/.gitconfig` 指令
2. 在其中输入

```
1 [alias]
2 graph = log --all --graph --decorate --oneline
```

3. 运行结果如下图 6

```
[alias]
graph = log --all --graph --decorate --oneline
```

图 6 通过 vim 来编辑.gitconfig 文件

```
Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.io (master)
$ vim ~/.gitconfig

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/missing-semester-cn.g
ithub.io (master)
$ git graph
* af054fa (HEAD -> master, origin/master, origin/HEAD) Merge pull request #172
from pspdada/master
|
| * 9baa48c remove irrelevant text
| * f5df7de fix wrong index
| * ef9a2f7 fix typo
|/
| * dd3f3dd Merge pull request #171 from HowieChih/for-better-understanding
|/
| * 8e26b4a 更新#课程概览与 shell##一个功能全面又强大的工具关于修改亮度文件报错
的翻译
| * 4e2ff43 更新#课程概览与 shell##一个功能全面又强大的工具关于修改亮度文件报错
的翻译
|/
| * d284d3e Merge pull request #170 from crosscap/typo-fix
|/
```

图 7 起了别名后的运行结果

2. 实际运用

1. 连接 github

- 我已经连接过了 github, 连接之后.gitconfig 文件中会有如图 8 的内容, 连接的指令如下

```
1 git config --global user.name "name"//自定义用户名
2 git config --global user.email "youxiang@qq.com"//用户邮箱
```

```
[user]
email = Danmushu@outlook.com
name = Danmushu
```

图 8 配置了过后的.gitconfig 文件

2. 建立本地版本库

- 使用 `git init` 建立, 如图 9

```
Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git init
Initialized existing Git repository in D:/knowledge/course/missingSemester/tes
t/.git/

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ ls -la
./ ../ .git/
```

图 9 git init

3. 查看当前状态

- 使用 `git status`, 如图 10

```

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ touch test.txt

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt

nothing added to commit but untracked files present (use "git add" to track)
Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)

```

图 10 git init

4. 把项目源代码加入仓库

- 使用 `git add .`，如图 11

```

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git add .
Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)

```

图 11 git add

5. 提交仓库

- 使用 `git commit -m "first commit"` 对起脚进行说明，如图 12

```

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git commit -m "first commit"
[master (root-commit) fdb4840] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git status
On branch master
nothing to commit, working tree clean

Danmu@Danmushu MINGW64 /d/knowledge/course/missingSemester/test (master)
$ git graph
* fdb4840 (HEAD -> master) first commit

```

图 12 git commit

6. 配置 ssh key

- git bash 中指令 `ssh-keygen -t rsa -C "youxiang@qq.com"` 会生成一对 ssh 密钥，将公钥放到 github 上（具体内容网上可查，不再赘述）

7. 推送

- 使用 `git push -u origin master`
 - 如果是 fork 的别人的仓库，可以发起 pull request 申请 merge
- git 实例一共 12 个

二、Latex

- 我已经使用过 latex 一段时间了，这里根据我之前用 latex 使用的实验报告作为实例

1. 基本结构

- 代码如下：

```

1 \documentclass[12pt]{article}
2
3 \usepackage{amsmath, amsthm, amssymb, color, latexsym}% 重要别删，对 newenvironment 有用
4 \usepackage[a4paper, left=1cm, right=1cm, top=0.5cm, bottom=1cm]{geometry}
5 \usepackage[all]{xy}
6 \usepackage{geometry}
7 \geometry{letterpaper}
8 \usepackage{graphicx}
9 \usepackage{url}
10 \usepackage{ctex}
11 \usepackage[colorlinks=true, allcolors=blue]{hyperref}
12 \usepackage{type1cm}
13 \fontsize{10.5pt}{15.75pt}

```

```

14
15 \newtheorem{problem}{Problem}
16
17 \newenvironment{jielun}[1][\it\bf{结论}]{\textbf{#1. }}{\ }
18 \newenvironment{taolun}[1][\it\bf{讨论}]{\textbf{#1. }}{\ }
19
20 \pagestyle{empty}
21 %%%%%%%%%%%
22 \begin{document}
23 \noindent \CJKfamily{zhkai}class name\hfill 16 \\\
24 HW4. 202404\hfill 梁子毅
25
26 % 顶格 线
27 \noindent\hrulefill
28 \CJKfamily{zhkai}
29 \begin{problem}
30
31 \end{problem}
32 \begin{enumerate}
33 \item
34 \end{enumerate}
35
36 \begin{quote}
37
38 \end{quote}
39
40 \begin{problem}
41
42 \end{problem}
43
44
45 \end{document}
46

```

代码解释

1. `\documentclass[12pt]{article}` : 这行代码定义了文档的类型为 `article`，并且字体大小设置为 12pt。
2. `\usepackage` : 这些行引入了多个 LaTeX 包，用于提供额外的功能，比如数学公式、定理环境、颜色、图形插入等。
3. `\geometry` : 这个命令用于设置页面的尺寸和边距。
4. `\newtheorem` : 定义了一个名为 `problem` 的新定理环境，用于标记问题。
5. `\newenvironment` : 定义了两个新的环境 `jielun` 和 `taolun`，分别用于标记结论和讨论部分。
6. `\pagestyle{empty}` : 设置页面的样式为无页眉和页脚。
7. `%` : 是注释的符号，我这里用来作为分隔符。
8. `\begin{document}` : 这标志着文档内容的开始。
9. `\noindent` : 这行代码防止段落首行缩进。
10. `\CJKfamily{zhkai}` : 这行代码指定了使用中文楷体字体。
11. `class name\hfill 16 \\\` : 这行代码设置了文档的标题和日期，`hfill` 用于水平填充空白，使日期右对齐。
12. `HW4. 202404\hfill 梁子毅` : 这可能是作业的编号和作者的名字，同样使用 `hfill` 进行右对齐。
13. `\noindent\hrulefill` : 这行代码在文档中添加了一条水平线。
14. `\begin{problem}` : 开始一个新的问题环境。
15. `\begin{enumerate}` : 开始一个枚举列表。
16. `\end{enumerate}` : 结束枚举列表。
17. `\begin{quote}` : 开始一个引用环境。
18. `\end{quote}` : 结束引用环境。

- 19. `\begin{problem}` : 开始另一个问题环境。
- 20. `\end{document}` : 这标志着文档内容的结束。

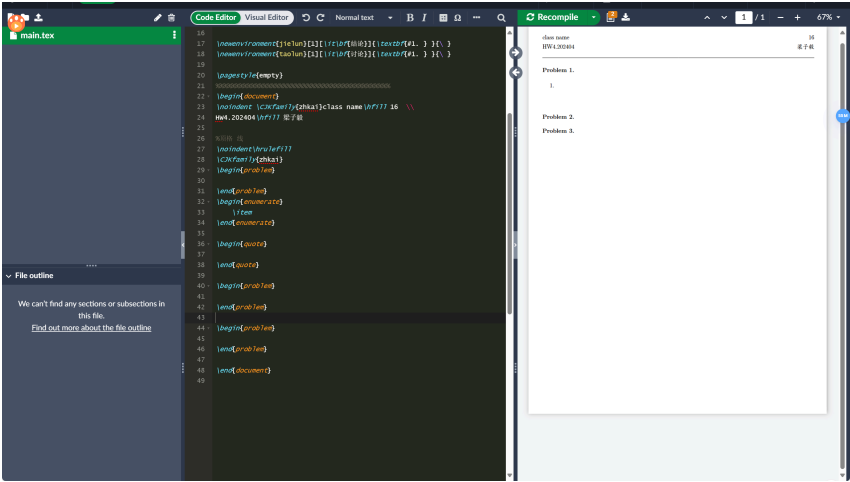


图 13 我的一份 latex 模板

- 除此之外，常用的还有 `\section` , `\title` , `\author`
- 如下图 14 是我上个学期使用的一份报告模板

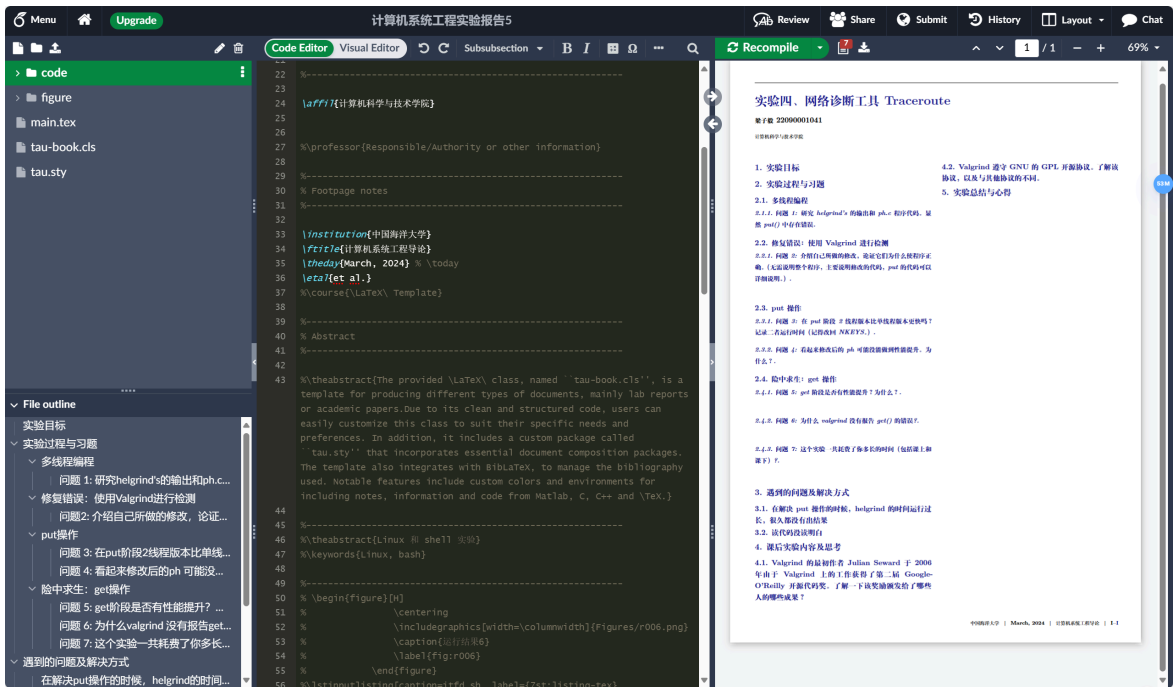


图 14 我的一份 latex 模板

- Latex 实例一共 20 个（20 个命令）

三、 总结

- Git
 - 1. git 工具的思想相当有意思，但是接口的设计相当“丑”，很难记，我上 github 看了 git 的源码，感觉文件结构很混乱（大概是因为我功力不到家），看不透。
 - 2. 目前我使用 git 会用 clion 等 jetbrain 的 ide 中的简便图形界面，当连接好仓库之后，直接点击图形界面的 commit 和 push 还是相当方便的

- Latex
 1. 上一个学期用了大半个学期的 latex 来写实验报告，中途挑选了一个 tau-book 的模板图 14 这个模板比较好看
 2. 使用 latex 的感觉就是这个工具又臭又长，非常难用，但是确实比 markdown 的编译效果美观很多
 3. 现在使用写报告用的是 typst，一个新的排版语言，底层用 rust 构建，编译速度很快，而且很好配置（latex 我配置了大半天都还是报错，最后使用 overleaf 来写）
 4. 总的来说，如果没有一定的必要，推荐使用 typst 这个新生代来进行编写实验报告或者笔记，工具玩来玩去，轻量和便捷慢慢成为追求，满足需求即是王道