

系统开发工具基础实验报告（一）

姓名：陈怡冰 学号：23020007007

August 27, 2024

目录

1	课程概览与版本控制	2
1.1	实验内容	2
1.2	实验结果	2
1.3	实验代码及关键步骤	11
1.4	实验中遇到的问题及解决方案	16
1.4.1	合并时发生冲突	16
1.4.2	无法克隆仓库	16
1.4.3	全局配置忽略规则失败	17
1.4.4	合并时出现冲突	17
2	LaTeX 文档编辑	18
2.1	实验内容	18
2.2	实验结果	18
2.3	实验代码及关键步骤	20
3	实验心得	23
3.1	课程概览与版本控制	23
3.2	LaTeX 文档编辑	23
4	我的 Github 仓库网址	24

实验部分 1 课程概览与版本控制

1.1 实验内容

1. 阅读学习笔记 & 观看讲座视频，了解 Git 的基本结构。
2. 创建分支。
3. 删除分支。
4. 将分支中的 python 文件进行融合。
5. 克隆本课程网站的仓库。
6. 查看文件是否被修改。
7. 将版本历史可视化并进行探索：
8. 是谁最后修改了 **README.md** 文件？（提示：使用 **git log** 命令并添加合适的参数）
9. 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**）
10. 使用 Git 时的一个常见错误是提交本不应该由 Git 管理的大文件，或是将含有敏感信息的文件提交给 Git。尝试向仓库中添加一个文件并添加提交信息，然后将其从历史中删除。
11. 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？当您执行 **git log -all -oneline** 时会显示什么？
12. 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？
13. 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；
14. 您可以通过执行 **git config --global core.excludesfile /.gitignore_global** 在 **/.gitignore_global** 中创建全局忽略规则。配置您的全局 **gitignore** 文件来自动忽略系统或编辑器的临时文件，例如 **.DS_Store**；

1.2 实验结果

1. 实例 1

Git 将目录和文件的结构看成树（与 Linux 系统的结构相契合）

```
1          <root> (tree)
2      |
3      +- foo (tree)
4          |
5          + bar.txt (blob, contents = "hello world")
6      |
7      +- baz.txt (blob, contents = "git is wonderful")
```

而历史记录是一个由快照组成的有向无环图【所以无法修改历史记录，只能删减或增加历史记录】，包含对象和引用的概念【可以用哈希值和名称来访问提交的历史记录 *or* 文件】。

```
daisy@daisy-virtual-machine:~/class$ git checkout master
M      cat.py
切换到分支 'master'
daisy@daisy-virtual-machine:~/class$ git branch -d cat
```

图 1.1: 实例 2 结果图

2. 实例 2

创建包含 Tea() 的分支 tea 和包含 Coffee() 的分支 Coffee。

```
daisy@daisy-virtual-machine:~/class2$ git log --all --graph --decorate
* commit a103f0aba87292c08e70416757ef65018b0a016e (coffee)
| Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
| Date:   Mon Aug 26 20:56:06 2024 +0800
|
|     main.py
|
| * commit 52e341d0bf779ef7ff19efd023dabe24c3bbe8b5 (HEAD -> tea)
| / Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
|   Date:   Mon Aug 26 20:51:11 2024 +0800
|
|     main.py
|
| * commit 6ff44b44cc2b51bbcbdfef0b9d605bfbfca9883d (master)
| Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
| Date:   Mon Aug 26 20:43:06 2024 +0800
|
|     main.py
```

图 1.2: 实例 2 结果图

3. 实例 3

使用 `git branch -b` 来删除分支【前提是 HEAD 不指向当前分支】

```

daisy@daisy-virtual-machine:~/class$ git checkout master
M      cat.py
切换到分支 'master'
daisy@daisy-virtual-machine:~/class$ git branch -d cat

```

图 1.3: 实例 3 结果图

4. 实例 4

将包含 Tea() 的分支 tea 和包含 Coffee() 的分支 Coffee 合并。

```

*   commit c4c9beac2182b8bfa989b1910076f3e286359177 (HEAD -> tea)
|\  Merge: 52e341d a103f0a
| | Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
| | Date:   Mon Aug 26 21:01:31 2024 +0800
| |
| |     main.py
| |
| *   commit a103f0aba87292c08e70416757ef65018b0a016e (coffee)
| | Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
| | Date:   Mon Aug 26 20:56:06 2024 +0800
| |
| |     main.py
| |
| *   commit 52e341d0bf779ef7ff19efd023dabe24c3bbe8b5
|/  Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
|   Date:   Mon Aug 26 20:51:11 2024 +0800
|
|   main.py

```

图 1.4: 实例 4 结果图

5. 实例 5

经过多次尝试终于成功克隆。（失败原因将在“1.4 实验中遇到的问题及解决方案中”列出）

```
daisy@daisy-virtual-machine:~$ git init
已重新初始化已存在的 git 仓库于 /home/daisy/.git/
daisy@daisy-virtual-machine:~$ git clone https://github.com/missing-semester-cn/missing-semester-cn.git
正克隆到 '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)
接收对象中: 100% (3194/3194), 15.44 MiB | 5.79 MiB/s, 完成.
处理 delta 中: 100% (2040/2040), 完成.
```

图 1.5: 实例 5 结果图

6. 实例 6

查看重新编辑后的文件和 HEAD 指向的历史记录中的文件的差异。

```
daisy@daisy-virtual-machine:~/class$ git checkout master
M      cat.py
切换到分支 'master'
daisy@daisy-virtual-machine:~/class$ git branch -d cat
```

图 1.6: 实例 6 结果图

7. 实例 7

用了两种方法对版本历史可视化进行修饰。

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git log --all --graph --decorate --oneline
* 9d72e1f (master) modify some words
* 4ce1f05 add hello.txt
* d50a857 (HEAD, origin/master, origin/HEAD) Merge pull request #172 from pspd/ada/master
| \
| * 1eecd59 remove irrelevant text
| * 0f90ef1 fix wrong index
| * 12802cd fix typo
| /
* 7534662 Merge pull request #171 from HowieChih/for-better-understanding
| \
| * 26fce64 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
| * ad0fbe8 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
| /
* a965910 Merge pull request #170 from crosscap/typo-fix
```

图 1.7: 实例 7(1) 使用 “--oneline” 进行可视化修饰的结果图

```

*   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
:

```

图 1.8: 实例 7(2) 没有使用 “-online” 进行实例化的结果图

8. 实例 8

由图可知，最后修改 README.md 的人是 [yuzq <yuzq@sunwayworld.com>](mailto:yuzq@sunwayworld.com)

```

daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ 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 1.9: 实例 8 结果图

9. 实例 9

由图可知，最后一次 **collections:** 行的提交信息是 [Redo lectures as a collection](#)


```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git show --pretty=
format:"%s" a88b4eac | head -1
Redo lectures as a collection
```

图 1.10: 实例 9 结果图

10. 实例 10

提交编写的 hello.txt 文件，然后从历史中删除（文件也同时删除了）

```
commit 9cfaf8028bef4bedc73ac28db0ab2afe6dffa7266 (HEAD -> master)
Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
Date:   Sun Aug 25 13:02:04 2024 +0800

    add a useless message

commit d50a857ac3ff8fa5d6da6ae101dd76bdb94e21ad (origin/master, origin/HEAD)
Merge: 7534662 1eecd59
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
```

图 1.11: 实例 10(1) 删除前的版本历史记录

```

* 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
:

```

图 1.12: 实例 10(2) 删除后的版本历史记录

11. 实例 11

使用 stash 保存当前的修改时，Git 会自动创建一个新的分支，并将 stash 的修改应用到这个新的分支上。然后，Git 会在此新分支上进行一次提交，并以 'WIP' 作为提交的消息。这个 'WIP' 提交其实就是保存了之前的工作进度，可以在需要的时候轻松地切换回这个分支，继续之前的工作，不会因为 checkout 而被销毁。

如果当前分支有修改，又不想提交或放弃，可以使用 `git stash` 将改动存到暂存区。

```

daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
log --all --oneline
6ecc576 (refs/stash) WIP on master: 4ce1f05 add hello.txt
3aed9d9 index on master: 4ce1f05 add hello.txt
4ce1f05 (HEAD -> master) add hello.txt

```

图 1.13: 实例 11 将 hello.txt stash 以后

12. 实例 12

用 `git stash pop` 即可将暂存区的改动恢复至当前分支。

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
log
commit 4ce1f055966ccaf4900d75baa79202b06b68dce7 (HEAD -> master)
Author: ChenFirstIce <chenyibing@stud.edu.ouc.cn>
Date:   Sun Aug 25 13:20:32 2024 +0800

    add hello.txt
```

图 1.14: 实例 12 结果图

13. 实例 13

创建别名之后可以直接用 `git graph` 命令输出 `git log -all -graph -decorate -oneline`。

```
daisy@daisy-virtual-machine:~$ cd missing-semester-cn.github.io
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
graph
* 4ce1f05 (HEAD -> master) add hello.txt
* d50a857 (origin/master, origin/HEAD) Merge pull request #172
from pspdada/master
|\
| * 1eecd59 remove irrelevant text
| * 0f90ef1 fix wrong index
| * 12802cd fix typo
|/
* 7534662 Merge pull request #171 from HowieChih/for-better-und
erstanding
|\
| * 26fce64 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
| * ad0fbe8 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
|/
* a965910 Merge pull request #170 from crosscap/typo-fix
```

图 1.15: 实例 13 结果图

14. 实例 14

通过执行 `git config --global core.excludesfile ~/.gitignore__global` 在 `~/.gitignore__global` 中创建全局忽略规则。配置全局 `gitignore` 文件来自动忽略系统或编辑器的临时文件，我这里用的是一个普普通通的 `hello.txt` 文件。[这里有一点问题，我使用了另一种方法。](#)

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
status
头指针分离于 d50a857
无文件要提交，干净的工作区
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
add hello.txt
下列路径根据您的一个 .gitignore 文件而被忽略：
hello.txt
```

图 1.16: 实例 14 结果图

1.3 实验代码及关键步骤

1. 一些常用的 git 相关命令。因为分支的名称和哈希值为引用与对象的关系，所以一般两者能够混在一起用

```
1 git init                #仓库初始化
2 git clone [url]         # 克隆一个远程仓库到本地目录，其中[url]是远程仓库
   的 URL
3 git add [文件名]        # 将指定文件添加到暂存区，准备提交。可以使用.代
   替[文件名]来添加所有修改过的文件
4 git commit -m "提交信息" # 将暂存区的更改提交到仓库中，并附上提交信息
5 git log                 # 查看提交历史记录
6 git log --all --graph -- decorate #使历史版本形成形象的图
7 git log --all --graph -- decorate --oneline #更简洁的图
8 git diff # 查看工作目录与暂存区之间的差异
9 git checkout [分支名称or分支哈希值] #改变分支
10 git branch [分支名称or分支哈希值] # 创建一个新分支，但不切换
11 git branch -d [分支名称or分支哈希值] # 删除指定分支，如果该分支已合并
12 git merge [分支名称or分支哈希值] # 将指将分支合并到当前分支
13 git remote             # 查看远程仓库
14 git remote add [远程仓库名称] [url] # 添加一个新的远程仓库
```

2. git log 查看提交历史，其中-1表示之查看最近一次提交。

```
1 git log -1 README.md
```

3.
 - git blame __config.yml: 这个命令显示每一行最后一次被修改的详细信息，包括提交的哈希值、作者和日期。
 - grep collections: 这个命令用来过滤输出，只显示包含“collections”这个词的行。

```
1 git blame _config.yml | grep collections
```

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git blame _config.
yml | grep collections
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 18) collections:
```

图 1.17: 查看提交时 collection 行的哈希值

- `git show -pretty=format:%s a88b4eac`: 这个命令会显示提交 a88b4eac 的提交信息摘要 (%s 只显示提交信息的标题部分)。
- | `head -1`: 这个部分将只显示输出的第一行。

```
1 git log --pretty=format:"%s" a88b4eac -1
```

图 1.18: 显示提交信息摘要的第一行

4. 提交一个 hello.txt 的文件，可以用 ls 查看仓库中的文件，其中有 hello.h

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ ls
_2019          favicon-16x16.png  index.md
_2020          favicon-32x32.png  _layouts
404.html       favicon.ico        lectures.html
about.md       Gemfile            license.md
apple-touch-icon.png Gemfile.lock       README.md
CNAME          hello.txt          robots.txt
_config.yml    _includes          static
```

图 1.19: 删除前仓库的文件 list

- 使用命令删除提交，同时文件也被删除了。

```
1 git filter-branch --force --index-filter 'git rm --cached --ignore
  -unmatch hello.txt' \ --prune-empty --tag-name-filter cat -- --
  all
```

```
* commit d50a857ac3ff8fa5d6da6ae101dd76bdb94e21ad (HEAD -> master, origin/master, origin/HEAD)
|\ Merge: 7534662 1eecd59
| | 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 1eecd59e606f55f7f636ecadf9779b5eca367eaf
```

图 1.20: 使用命令后版本历史

5. 删除前仓库中存在 hello.txt 的文件。

- 使用命令

```
1 git filter-branch --force --index-filter '\git rm --cached --ignore-unmatch hello.txt' \--prune-empty --tag-name-filter cat -- --all
```

- 历史提交记录中不再有 hello.txt 的提交历史。

```
* commit d50a857ac3ff8fa5d6da6ae101dd76bdb94e21ad (HEAD -> master, origin/master, origin/HEAD)
|\ Merge: 7534662 1eecd59
| | 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 1eecd59e606f55f7f636ecadf9779b5eca367eaf
```

图 1.21: 使用命令后的提交历史

6.
 - 使用 git stash 会保存当前的进度，使文件处于 WIP 状态中，checkout 不会销毁进度。

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
stash
保存工作目录和索引状态 WIP on master: 4ce1f05 add hello.txt
```

图 1.22: 文件处于 WIP 状态

- 版本历史中会多出一个暂时的分支。

```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
log --all --oneline
6ecc576 (refs/stash) WIP on master: 4ce1f05 add hello.txt
3aed9d9 index on master: 4ce1f05 add hello.txt
4ce1f05 (HEAD -> master) add hello.txt
```

图 1.23: git log 查看版本历史

7. 使用 git stash pop 会销毁暂时的缓冲区。

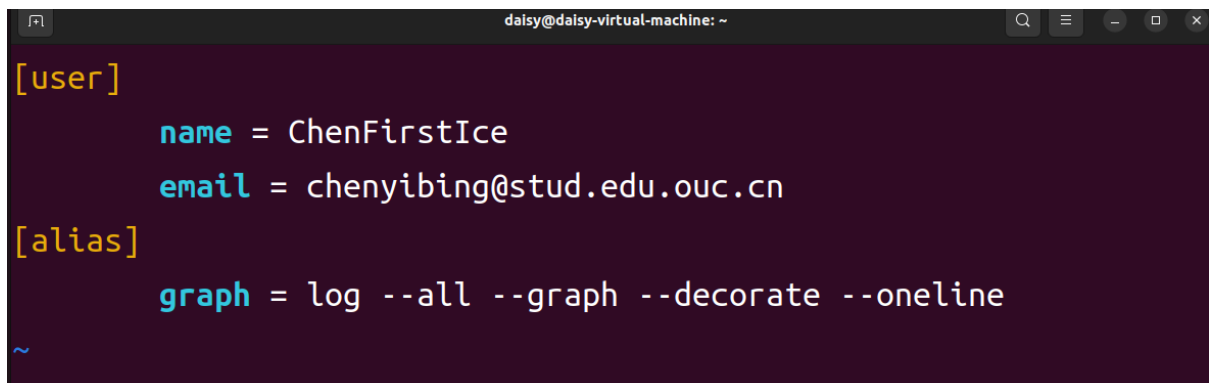
```
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
stash pop
位于分支 master
您的分支领先 'origin/master' 共 1 个提交。
(使用 "git push" 来发布您的本地提交)

尚未暂存以备提交的变更:
(使用 "git add <文件>..." 更新要提交的内容)
(使用 "git restore <文件>..." 丢弃工作区的改动)
    修改:      hello.txt

修改尚未加入提交 (使用 "git add" 和/或 "git commit -a")
丢弃了 refs/stash@{0} (6ecc5762a23976e1126094ddc01903cee78f8f03)
```

图 1.24: refs/stash 被销毁

8.
 - 使用 vim 编辑 ~/.gitconfig



```
daisy@daisy-virtual-machine: ~
[user]
    name = ChenFirstIce
    email = chenyingbing@stud.edu.ouc.cn
[alias]
    graph = log --all --graph --decorate --oneline
~
```

图 1.25: 增加别名

- 然后再使用 git graph 会出现和 git log --all --graph --decorate --oneline 一样的效果。

```

daisy@daisy-virtual-machine:~$ cd missing-semester-cn.github.io
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
graph
* 4ce1f05 (HEAD -> master) add hello.txt
* d50a857 (origin/master, origin/HEAD) Merge pull request #172
from pspdada/master
|\
| * 1eecd59 remove irrelevant text
| * 0f90ef1 fix wrong index
| * 12802cd fix typo
|/
* 7534662 Merge pull request #171 from HowieChih/for-better-und
erstanding
|\
| * 26fce64 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
| * ad0fbe8 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
|/
* a965910 Merge pull request #170 from crosscap/typo-fix

```

图 1.26: 效果图

9.
 - 将 hello.txt 在全局中忽略，再通过 git status 查看或者用 git add 没有效果。使用 `git config --global core.excludesfile /dev/null` 有效果的前提是 **hello.txt 未被跟踪**

```

daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
status
头指针分离于 d50a857
无文件要提交，干净的工作区
daisy@daisy-virtual-machine:~/missing-semester-cn.github.io$ git
add hello.txt
下列路径根据您的一个 .gitignore 文件而被忽略：
hello.txt

```

图 1.27: hello.txt 被忽略

1.4 实验中遇到的问题及解决方案

1.4.1 合并时发生冲突

。

```
<<<<<<< HEAD
def Tea():
    print("TeaTeaTea,yummy!")
=====
def Coffee():
    print("Oh, poor guy! You must be reduced to do all the teamwork only by your
self!")
>>>>>> coffee
```

图 1.28: 标注处的冲突

- 解决方案
文件会被标注出有冲突的部分，需要手动修改。

```
daisy@daisy-virtual-machine:~/class2$ python3 main.py
What do you want to drink, tea or coffee?coffee
Oh, poor guy! You must be reduced to do all the teamworke only by yourself
```

图 1.29: 合并后的结果

1.4.2 无法克隆仓库

- 无法克隆仓库

```
daisy@daisy-virtual-machine:~/桌面$ git clone https://github.com/missing-semester-cn/missing-semester-cn.github.io.git
正在克隆到 'missing-semester-cn.github.io'...
remote: Enumerating objects: 3194, done.
remote: Counting objects: 100% (3194/3194), done.
remote: Compressing objects: 100% (1126/1126), done.
error: RPC 失败。curl 92 HTTP/2 stream 0 was not closed cleanly: CANCEL (err 8)
error: 预期仍然需要 39 个字节的正文
fetch-pack: unexpected disconnect while reading sideband packet
fatal: 过早的文件结束符 (EOF)
fatal: fetch-pack: 无效的 index-pack 输出
```

图 1.30: 无法克隆

- 解决方案
是虚拟机没有连上客机的网络，调整虚拟机的网络连接方式为**桥连模式**即可。



图 1.31: 虚拟机网络连接方式

1.4.3 全局配置忽略规则失败

- 使用命令 `git config --global core.excludesfile ~/.gitignore__ global hello.txt` 失败，hello.txt 文件并没有被忽略。
- 解决方案**
直接用 vim 命令修改 `~/.gitignore__ global`，输入 hello.txt。此时 hello.txt 被忽略。

```
daisy@daisy-virtual-machine:~/class$ vim ~/.gitignore_glob
al
```

图 1.32: 修改忽略规则文件

1.4.4 合并时出现冲突

- 合并时会出现不可以自动修复的冲突，文件中会标出冲突的部分，需要自行人工修改。

```
<<<<<<< HEAD
def Tea():
    print("TeaTeaTea,yummy!")
=====
def Coffee():
    print("Oh, poor guy! You must be reduced to do all the teamwork only by your
self!")
>>>>>> coffee
```

图 1.33: 文件中标出分支之间的冲突

- 解决方案**
手动更改。

实验部分 2 LaTeX 文档编辑

2.1 实验内容

1. 改变字体颜色。
2. 改变有序列表整体的序号形式。
3. 在 XeTeX 编辑器下打出中文。
4. 显示 Shell 命令（语言为 bash）。
5. 在 LaTeX 中编写伪代码。
6. 在 LaTeX 中插入图片。
7. 设置图片在指定位置。

2.2 实验结果

1. **实例 1**
将 *README.md* 变成蓝色粗体字。



图 2.1: 实例 1 结果图

2. **实例 2**
将序号全部改为阿拉伯数字。

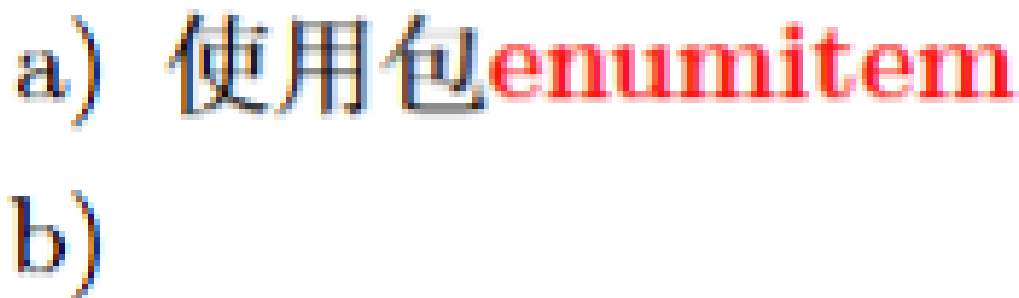


图 2.2: 实例 2 结果图

3. **实例 3**
不再做图片展示。

4. [实例 4](#)
详情请见 2.3 实验代码及关键步骤中实例 1 和实例 2 的代码展示部分。
5. [实例 5](#)
制作三线表格式伪代码图。

Fuction 2 computeTax
Input: <i>Rule rules[], int n, int income</i>
Output: <i>sum</i>
1: <i>sum</i> \leftarrow 0
2: if <i>income</i> > <i>rules</i> [<i>n</i>]. <i>M</i> then
3: <i>sum</i> \leftarrow <i>sum</i> + (<i>income</i> - <i>rules</i> [<i>n</i>]. <i>M</i>) * <i>rules</i> [<i>n</i>]. <i>R</i> * 0.01
4: <i>income</i> \leftarrow <i>rules</i> [<i>n</i>]. <i>M</i>
5: end if
6: for each <i>i</i> \in [<i>n</i> , 1] do
7: if <i>income</i> <= <i>rules</i> [<i>i</i>]. <i>M</i> and <i>income</i> > <i>rules</i> [<i>i</i> - 1]. <i>M</i> then
8: <i>sum</i> \leftarrow <i>sum</i> + (<i>income</i> - <i>rules</i> [<i>i</i> - 1]. <i>M</i>) * <i>rules</i> [<i>i</i> - 1]. <i>R</i> * 0.01
9: <i>income</i> \leftarrow <i>rules</i> [<i>i</i> - 1]. <i>M</i>
10: end if
11: end for

图 2.3: 实例 5 结果图

6. [实例 6](#)
不再做展示。

- 实例 7
使图片在相应的标题下，而会为了适应页面而乱跑。

2.2 实验结果

- 实例 1
将 *README.md* 变成蓝色粗体字。
- 实例 2
将序号全部改为阿拉伯数字。
- 实例 3
不再做图片展示。
- 实例 4
详情请见 2.3 实验代码及关键步骤中实例 1 和实例 2 的代码展示部分。
- 实例 5
三线表格式伪代码图。

2.3 实验代码及关键步骤

- 实例 1
使用



图 2.1: 实例 1 结果图

图 2.4: 实例 7 结果图

2.3 实验代码及关键步骤

- 实例 1
使用

```
1 {\textbf{\color{颜色}文字}}
```

- 实例 2

- 使用包 **enumitem**
- 使用代码

```
1 \begin{enumerate}[label=\arabic*.]
2 \item 第一项
3 \item 第二项
4 \item 第三项
5 \end{enumerate}
```

其中:

label=\arabic*. 表示使用阿拉伯数字加点的形式 (例如 1.、2.、3.)。
label=\alph* 表示使用小写字母加括号的形式 (例如 a)、b)、c))。

3. 实例 3

- a) 使用`xeCJK`包
- b) 在`\begin{document}`前使用

```
1 \setCJKmainfont{字体名称}
```

可以改变全文字体。

4. 实例 4

- a) 使用`lstlisting`包
- b) 将语言改为`bash`即可。

```
1 \begin{lstlisting}[language=语言]
```

5. 实例 5

- a) 使用包`algorithmic`和`algorithm`来排版伪代码。
- b) 命令`\STATE`表示 1 个语句并用`$ $`里括住伪代码的内容。`\IF`和`\ENDIF`等与此相同。
- c) 代码如下：

```
1 \def\SetClass{article}
2 \documentclass{\SetClass}
3 \usepackage{algorithmic}
4 \usepackage{algorithm}
5
6 \begin{document}
7
8 \floatname{algorithm}{Function}
9 \begin{algorithm}[!h]
10 \caption{computeTax}
11 \renewcommand{\algorithmicrequire}{\textbf{Input:}}
12 \renewcommand{\algorithmicensure}{\textbf{Output:}}
13 \begin{algorithmic}[1]
14 \REQUIRE $Rule\sim rules[],\sim int\sim n,\sim int\sim income$
15 \ENSURE $sum$
16 \STATE $sum \leftarrow 0$
17 \IF{$income > rules[n].M$}
18 \STATE $sum \leftarrow sum + (income - rules[n].M) \times rules[n].R \times 0.01$
19 \STATE $income \leftarrow rules[n].M$
20 \ENDIF
21 \FOR{each $i$ \in $[n, 1]$}
22 \IF{$income \leq rules[i].M$ \AND $income > rules[i-1].M$}
23 \STATE $sum \leftarrow sum + (income - rules[i-1].M) \times rules[i-1].R \times 0.01$
24 \STATE $income \leftarrow rules[i-1].M$
25 \ENDIF
26 \ENDFOR
27 \end{algorithmic}
28 \end{algorithm}
29
30 \end{document}
```

6. 实例 6

- a) 将需要插入的图片放入同一个 project 的 images（自行创建）文件夹中。

b) 使用

```
1 \begin{figure}  
2 \centering  
3 \includegraphics[width=\textwidth]{}  
4 \caption{Caption}  
5 \label{fig:my_label}  
6 \end{figure}
```

在`\includegraphics[width=\textwidth]{}{}`最后的中括号中输入图片名即可。

7. 实例 7

使用`float`包，并在`\begin{figure}`后设置[H]模式

实验部分 3 实验心得

3.1 课程概览与版本控制

- 通过学习老师提供的资料，我对 Git 这一版本控制工具有了更加全面和深入的理解。课程不仅帮助我巩固了基础知识，还引导我探索了一些高级操作和实际应用场景，使我在项目管理和团队协作中更为自信。
- 课程版本控制（Git）课程笔记中对 ‘HEAD’ 指针的深入探讨和对 Git 内部结构的剖析清晰易懂，让我真正理解了**对象**和**引用**之间的关系，并让命令的记忆更加深刻。
- 我学会了如何有效利用 Git 的历史查看来管理版本历史。这让我更加理解了 Git 在大规模项目中的优势，尤其是在不同分支之间快速切换时的高效性。此外，通过实践各类实例，我学会了如何保持提交历史的整洁和线性，从而提高了项目的可维护性和可读性。

3.2 LaTeX 文档编辑

- 在 LaTeX 文档编辑实验中，我深入体验了 LaTeX 的强大功能和其在学术写作中的应用优势，真正做到**所想即所得**。LaTeX 的文档结构非常严谨，尤其是在排版复杂文档时展现出极高的效率。通过实验，我掌握了如何使用 LaTeX 的基本命令来创建结构化的文档，包括章节划分、列表生成以及文本格式化等。此外，LaTeX 在处理数学公式方面的能力也令人印象深刻，使用简单的语法便能生成专业且美观的数学表达式。这对于撰写涉及大量数学内容的论文和报告尤为重要。
- 总的来说，LaTeX 虽然学习曲线较为陡峭，但通过实验，我感受到它在处理复杂文档时的无与伦比的优势，特别是在学术领域，它不仅提高了排版的质量，还规范了写作流程，是值得深入学习和掌握的工具。

附录 1

我的 Github 仓库网址

我的 Github 仓库 [\[点击即可\]](#)