

第一次实验报告

时伟杰

2024 年 8 月 23 日

目录

1	git 简介	1
2	Latex 简介	1
3	实例演示与截图	2
4	心得体会	3

1 git 简介

Git 是一个免费的开源分布式版本控制系统，旨在快速高效地处理从小型到大型项目的所有事务。它通过记录每次文件的改动，允许同事协作编辑，从而简化了文件管理过程。Git 能够自动记录每次文件的改动，使得查看某次改动变得简单，同时支持离线开发，速度和性能优异，特别适合大型项目的多人协作开发。由于其分布式特性、暂存区设计和出色的分支系统，Git 能够灵活应对各种工作场景，轻松实现不同工作模式的切换。此外，Git 拥有庞大的用户社区和丰富的资源，使得学习和获取帮助变得相对容易。由于其灵活性和受欢迎程度，Git 已成为所有团队的绝佳选择，被广泛用于项目管理、代码跟踪和团队协作中。

2 Latex 简介

LaTeX 是一种基于 TEX 的排版系统，它利用纯文本格式进行文档排版，即使使用者没有排版和程序设计的知识，也能充分发挥由 TeX 所提供的强大功能，在几天甚至几小时内生成具有书籍质量的印刷品。LaTeX 特别适用于生成高印刷质量的科技和数学类文档，也适用于从简单的信件到完整书籍的所有其他种类的文档。它的优点包括专业排版、数学支持、跨平台性、开源免费、分章节管理和引用管理等功能。LaTeX 采用了一种基于标记的方式来创建文档，这种方式允许用户更好地控制文档的排版和格式，与常见的文字处理软件如 Microsoft Word 不同。LaTeX 文档通常包括文档类定义、导言部分（包含设置文档格式、加载宏包和定义自定义命令的内容）、正文部分（包含文档的实际内容，如文字、图像、表格和公式）、章节以及公式和图像的插入。此外，LaTeX 提供了功能强大的数学排版功能，支持行内和行间数学公式，以及插入图像并控制其位置和大小。

3 实例演示与截图

git 操作	实例	介绍
1	git config --global user.name	向 git 提交姓名
2	git config --global user.email	向 git 提交邮箱
3	date	日期
4	git init	创建一个本地的项目仓库
5	git add 123.txt	将 123.txt 提交到暂存区
6	git commit -m "这是第一次"	将文件提交到仓库
7	git log	查看提交的节点哈希值以及时间和作者
8	git log --stat	比 git log 多了文件内容变化
9	git checkout	代码回溯
10	git branch	查看分支
11	git checkout -b develop	创建并转化到新建分支
12	git merge develop	合并分支
Latex 操作	实例	介绍
13	documentclass	必须出现在每个 LaTeX 文档的开头
14	begin{document}	开头
15	end{document}	结尾
16	maketitle	制作题目
17	section{...}	节
18	subsection{...}	小节
19	chapter{...}	章
20	label{labelname}	标签
21	tableofcontents	创建目录
22	pagenumbering{...}	页码
23	newpage	新建页面
24	noindent	段落顶格
25	enumerate	有序列表
26	百分号	注释
27	反斜杠	特殊字符的转义字符
28	hline	插入一个贯穿所有列的横着的分割线

续表:

29	<code>begin{tabular} end{tabular}</code>	表格
30	<code>l</code>	一个左对齐的列
31	<code>r</code>	一个右对齐的列
32	<code>c</code>	一个向中对齐的列
33	<code> </code>	一个列的竖线
34	<code>begin{figure} end{figure}</code>	图表
35	<code>centering</code>	将图片放置在页面的中央
36	<code>includegraphics{...}</code>	自动将图放置到文档中
37	<code>begin{equation} end{equation}</code>	公式
38	<code>^</code> <code>_</code>	上下标
39	<code>frac{numerator}{denominator}</code>	分数
40	<code>sqrt{...}</code>	根号
41	<code>sum</code>	求和
42	<code>int</code>	积分

4 心得体会

Git 是一个分布式版本控制系统,能够高效地管理和记录代码变更。它的分支功能允许你在不影响主线的情况下进行实验和开发。GitHub 作为 Git 的平台,提供了远程存储、代码审查和团队协作的功能。可以使用 pull requests 进行代码审查,通过 issues 跟踪任务和问题。LaTeX 是一个强大的排版系统,特别适合排版数学公式和学术文档。它提供了精确的格式控制和自动化的参考文献管理,能帮助你创建高质量的文档。对于这些工具,实际操作和实践非常重要,使用模板和查阅社区支持也能加速学习进程。

在学习的过程中,也不是一帆风顺的。比如 git 如果想要在 Win11 桌面或者其他文件夹运行需要在注册表编辑器上编辑一些参数,经过网上大量的检索和学习才掌握。而且 Latex 的语法多而杂,在编辑时常常遇到一些问题,查询网上博客才能解决。这些问题也强化了我检索信息和解决问题的能力,提升了我的工作效率和文档质量,让我在技术项目和学术写作中更加得心应手。

插图

1	第一和二个实例演示	4
2	第三个实例演示	4
3	第四个实例演示	4
4	第五和六个实例演示	5
5	第七和八个实例演示	5
6	第九个实例演示	5
7	第十个实例演示	6
8	第十一个实例演示	6
9	第十二个实例演示	6

```
m1585@Ü MINGW64 / (master)
$ git config --global user.name"wj"

m1585@Ü MINGW64 / (master)
$ git config --global user.email"wj@163.com"
```

图 1: 第一和二个实例演示

```
m1585@Ü MINGW64 / (master)
$ date
Sun Sep  1 09:27:23      2024
```

图 2: 第三个实例演示

```
m1585@Ü MINGW64 /d/桌面
$ git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in D:/桌面/.git/
```

图 3: 第四个实例演示

```

m1585@Ü MINGW64 /d/桌面 (master)
$ git add 123.txt

m1585@Ü MINGW64 /d/桌面 (master)
$ git commit -m"这是第一次"
[master (root-commit) dc554e1] 这是第一次
1 file changed, 1 insertion(+)
create mode 100644 123.txt

```

图 4: 第五和六个实例演示

```

m1585@Ü MINGW64 /d/桌面 (master)
$ git log
commit dc554e1bd534a1c0b7a3ae69c0be14c0f1d11ac6 (HEAD -> master)
Author: wj <wj@163.com>
Date: Sun Sep 1 10:28:08 2024 +0800

    这是第一次

m1585@Ü MINGW64 /d/桌面 (master)
$ git log --stat
commit dc554e1bd534a1c0b7a3ae69c0be14c0f1d11ac6 (HEAD -> master)
Author: wj <wj@163.com>
Date: Sun Sep 1 10:28:08 2024 +0800

    这是第一次

123.txt | 1 +
1 file changed, 1 insertion(+)

```

图 5: 第七和八个实例演示

```

m1585@Ü MINGW64 /d/桌面 (master)
$ git checkout dc554e1bd534a1c0b7a3ae69c0be14c0f1d11ac6
M      123.txt
Note: switching to 'dc554e1bd534a1c0b7a3ae69c0be14c0f1d11ac6'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at dc554e1 这是第一次

```

图 6: 第九个实例演示

```
m1585@Ü MINGW64 /d/桌面 ((dc554e1...))
$ git branch
* (HEAD detached at dc554e1)
  master
```

图 7: 第十个实例演示

```
m1585@Ü MINGW64 /d/桌面 ((dc554e1...))
$ git checkout -b develop
Switched to a new branch 'develop'

m1585@Ü MINGW64 /d/桌面 (develop)
$ git branch
* develop
  master
```

图 8: 第十一个实例演示

```
m1585@Ü MINGW64 /d/桌面 (master)
$ git merge develop
Updating dc554e1..3bc7b63
Fast-forward
 123.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

图 9: 第十二个实例演示