# 系统开发工具基础实验报告

姓名：应佳钰
学号：22090022044
专业：计算机科学与技术

2025 年 9 月 9 日

# 目录

# 1 实验目的

- 掌握 Git 版本控制系统的基本操作和工作流程
- 学习 Git 分支管理策略和操作技巧
- 熟悉 LaTeX 文档编写基础，能够创建专业的技术文档
- 掌握 SSH 密钥的生成和配置方法，实现与 GitHub 的安全连接

# 2 实验环境

- 操作系统：Windows 11
- Git 版本：git version 2.51.0.windows.1
- 远程仓库平台：GitHub
- 命令行工具：PowerShell 5.1.26100.4768

# 3 实验原理

Git 分布式版本控制：采用快照方式记录文件变化，每个提交包含完整文件状态，通过哈希值唯一标识

SSH 非对称加密：使用公钥-私钥对实现安全认证，公钥存储在服务器，私钥保留在本地

LaTeX 文档编译：通过编写标记代码，经编译器处理生成高质量 PDF 文档，分离内容与格式

分支管理策略：使用功能分支进行并行开发，通过合并操作集成到主分支

# 4 实验内容与步骤

## 4.1 实验内容

- Git 基础操作与工作流程
- Git 分支管理与合并策略
- SSH 密钥生成与 GitHub 配置
- LaTeX 文档结构与元素编写
- LaTeX 表格、图片与数学公式插入

## 4.2 实验步骤

（1）Git 基础操作：

1. 配置全局信息：

```
git config --global user.name "yingjiayu"
git config --global user.email "12345678@qq.com"
```

这条命令只需在电脑上执行一次，它告诉 Git “我”是谁，这样之后的每次提交都会带有这些信息。

```
C:\Users\yingjiayu>git config --global user.name "yingjiayu"

C:\Users\yingjiayu>git config --global user.email "3128990996@qq.com"
```

2. 创建一个项目文件夹并进入：

```
mkdir my-first-git-project
cd my-first-git-project
```

```
C:\Users\yingjiayu>mkdir first-git-project

C:\Users\yingjiayu>cd first-git-project
```

3. 初始化一个新的 Git 仓库：

```
git init
```

命令会在当前目录创建一个新的、空的 Git 仓库。可以看到一个名为 `.git` 的隐藏文件夹，Git 用它来跟踪管理项目。

```
C:\Users\yingjiayu\first-git-project>git init
Initialized empty Git repository in C:/Users/yingjiayu/first-git-project/.git/
```

4. 创建文件并写入内容：

```
echo "# My First Git Project" > README.md
```

也可以用文本编辑器手动创建和编辑这个文件。

```
C:\Users\yingjiayu\first-git-project>echo "# My First Git Project" > README.md
```

5. 检查仓库状态：

```
git status
```

git status 命令显示工作目录和暂存区的状态。可以看到 README.md 被列为”Untracked files”（未跟踪的文件），意味着 Git 还没有开始跟踪这个文件的变化。

```
C:\Users\yingjiayu\first-git-project>git status
On branch master

No commits yet

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

6. 添加文件到暂存区：

```
git add README.md
# 或者添加所有变化文件
git add .
```

git add 命令将文件的变化“添加”到暂存区。暂存区是准备下次提交的内容。

```
C:\Users\yingjiayu\first-git-project>git add README.md
```

7. 再次检查状态：

```
git status
```

现在会看到 README.md 被列为”Changes to be committed”（要提交的更改），表示它已暂存。

```
C:\Users\yingjiayu\first-git-project>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
```

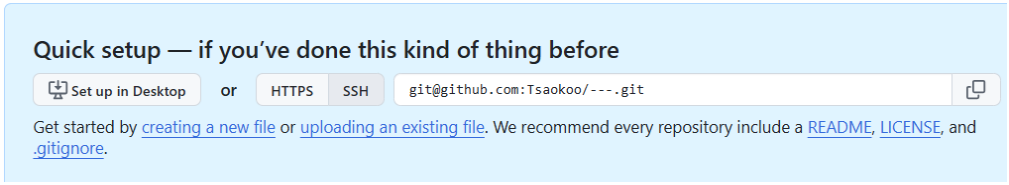
8. 将暂存区的更改提交到本地仓库：

```
git commit -m "Add README file"
```

git commit 命令将暂存区的所有更改永久地保存到本地仓库的历史记录中。-m 参数后面跟的是本次提交的说明，这个说明非常重要，应该清晰简洁地描述这次提交做了什么。

```
C:\Users\yingjiayu\first-git-project>git commit -m "Add README file"
[master (root-commit) bd74f9f] Add README file
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

9. 添加远程仓库：在 GitHub/Gitee 上创建一个新的空仓库。记下提供的远程仓库地址，可选 HTTPS 或 SSH 格式。




然后将本地仓库与远程仓库关联起来：

```
git remote set-url origin git@github.com:Tsaokoo/my-ouc-first-project.git
```

git remote add 命令为你的本地仓库添加一个名为 origin 的远程仓库地址。origin 是远程仓库的默认习惯命名。

```
C:\Users\yingjiayu\first-git-project>git remote add origin git@github.com:Tsaokoo/---.git
```

10. 推送到远程：

```
git push -u origin master
```

git push 命令将本地 master 分支（我是旧版本 Git r）上的所有提交推送到远程仓库 origin。-u 参数设置上游关联，这样以后直接使用 git push 即可。

（2）Git 分支操作：

1. 查看当前所有分支：

```
git branch
```

会看到只有一个分支 main (或 master)，并且当前就在这个分支上（前面有 * 号）。

```
C:\Users\yingjiayu\first-git-project>git remote add origin git@github.com:Tsaokoo/---.git
```

2. 创建一个新分支并切换到该分支：

```
git checkout -b new-feature
```

git checkout -b < 分支名 > 是 git branch < 分支名 >（创建分支）+ git checkout < 分支名 >（切换分支）的组合命令。现在是在一个名为 new-feature 的独立沙盒里工作，无论做什么都不会影响 main 分支。

```
C:\Users\yingjiayu\first-git-project>git branch
* master
```

3. 在新分支上做一些修改：

```
echo "This is a new feature." > feature.txt
git add feature.txt
git commit -m "Add a new feature file"
```

```
C:\Users\yingjiayu\first-git-project>git checkout -b new-feature
Switched to a new branch 'new-feature'
```

4. 切换回主分支 master：

```
git checkout master
```

查看文件列表，会发现 feature.txt 不见了，因为它只存在于 new-feature 分支上。这证明了分支的隔离性。

```
C:\Users\yingjiayu\first-git-project>echo "This is a new feature." > feature.txt

C:\Users\yingjiayu\first-git-project>git add feature.txt

C:\Users\yingjiayu\first-git-project>git commit -m "Add a new feature file"
[new-feature 4238c37] Add a new feature file
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

5. 将新分支的工作合并回主分支：

```
git merge new-feature
```

git merge 命令将指定分支 (new-feature) 的历史记录合并到当前所在分支 (main)。因为 new-feature 是从 main 直接分出去的，并且 main 自那以后没有新的提交，所以这是一个“快进合并”(Fast-forward)。现在再查看文件，feature.txt 就出现在 main 分支了。

```
C:\Users\yingjiayu\first-git-project>git checkout master
Switched to branch 'master'
```

6. 删除已经合并完毕的功能分支：

```
git branch -d new-feature
```

```
C:\Users\yingjiayu\first-git-project>git merge new-feature
Updating bd74f9f..4238c37
Fast-forward
 feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

7. 将本次合并的结果推送到远程仓库：

```
git push
```

（3）SSH 直连操作：

在之前的实验过程中，发现使用 HTTPS 会出现无法连接到 GitHub 服务器的情况，通常是由网络连接问题引起的，尤其是在中国内地访问 GitHub 时经常遇到。所以这里采用 SSH 替代 HTTPS。

1. 生成 SSH 密钥：

```
ssh-keygen -t rsa -b 4096 -C "1234567@qq.com"
```

按回车接受默认位置，然后直接按回车不设密码。

2. 将 SSH 密钥添加到 ssh-agent：我是 windows 操作系统，这里打开 PowerShell

```
# 确保 ssh-agent 服务已设置并启动
Get-Service ssh-agent | Set-Service -StartupType Automatic
Start-Service ssh-agent

# 添加你的 SSH 密钥
ssh-add ~\.ssh\id_rsa
```

```
PS C:\WINDOWS\system32> # 确保 ssh-agent 服务已设置并启动
>> Get-Service ssh-agent | Set-Service -StartupType Automatic
>> Start-Service ssh-agent
>>
```

```
>> # 添加你的 SSH 密钥
>> ssh-add ~\.ssh\id_rsa
~\.ssh\id_rsa: No such file or directory
PS C:\WINDOWS\system32> Get-Service ssh-agent | Set-Service -StartupType Automatic
PS C:\WINDOWS\system32> Start-Service ssh-agent
PS C:\WINDOWS\system32> ssh-add ~\.ssh\id_rsa
~\.ssh\id_rsa: No such file or directory
PS C:\WINDOWS\system32> # 检查 .ssh 目录是否存在
PS C:\WINDOWS\system32> Test-Path ~\.ssh
True
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> # 检查私钥文件是否存在
PS C:\WINDOWS\system32> Test-Path ~\.ssh\id_rsa
True
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> # 检查公钥文件是否存在
PS C:\WINDOWS\system32> Test-Path ~\.ssh\id_rsa.pub
True
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> # 列出 .ssh 目录中的所有文件
PS C:\WINDOWS\system32> Get-ChildItem ~\.ssh


    目录: C:\Users\yingjiayu\.ssh


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          2025/9/5      9:42           3381 id_rsa
-a----          2025/9/5      9:42            744 id_rsa.pub
-a----         2024/3/20     15:33            864 known_hosts
-a----         2024/3/20     15:32            104 known_hosts.old
```

3. 将公钥添加到 GitHub：显示公钥内容：

```
type %USERPROFILE%\.ssh\id_rsa.pub
```

复制输出的全部内容

登录 GitHub，进入 Settings → SSH and GPG keys → New SSH key

粘贴公钥内容并保存

4. 测试 SSH 连接：

```
ssh -T git@github.com
```

```
PS C:\WINDOWS\system32> # 使用完整路径添加 SSH 密钥
PS C:\WINDOWS\system32> ssh-add "$env:USERPROFILE\.ssh\id_rsa"
Identity added: C:\Users\yingjiayu\.ssh\id_rsa (3128990996@qq.com)
PS C:\WINDOWS\system32> # 确保 ssh-agent 服务已设置并启动
PS C:\WINDOWS\system32> Get-Service ssh-agent | Set-Service -StartupType Automatic
PS C:\WINDOWS\system32> Start-Service ssh-agent
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> # 添加你的 SSH 密钥
PS C:\WINDOWS\system32> ssh-add "$env:USERPROFILE\.ssh\id_rsa"
Identity added: C:\Users\yingjiayu\.ssh\id_rsa (3128990996@qq.com)
PS C:\WINDOWS\system32> # 显示公钥内容，复制全部输出
PS C:\WINDOWS\system32> Get-Content "$env:USERPROFILE\.ssh\id_rsa.pub"
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCuliSIOcc8O8XXQk3eHtp3Aj/KTWWgnmiLnGXIRtvg5P+b4Q6hVjl/UzajuKzh6pcfuJOwRQYTd/xYsYh2
0u4rb62q6r/+ElzmE6oFX+71Qlf5phYB4a51FV8mai+YQlfLHrLEX3MQgLtFOQJe5psoUlaFFhm/oakn1+GgsPBy2v6mkx5Otq5ejdkMirmKHk/SfVqJkNoQ
lfn7BTuariBrUc6vkMdotSSUSQHDoR7KSrmoE1nVMOmVQ1YtHOZhact+hQiRsA7TaD+JIWH9VcbxwahxWGRj7J7MPGafiUcOYOER1mK9cQtvVIDpPY1jW/Ae
mpnQHSmUNR7FV2KefCpDRHvTBnAm+tCPsBLFYgaZ9ieracNECIandmqkqW0ZI4zEUKsfYx3BuHipInVpTR23ZjJGKnMmp24+MkI+ihlCaT7yIhl6fXh1/cOO
4NASoSuM2dsZTT9tS+tkNeV8gqFsKOP+Rz+xOnNe2+OCMtQ7rn1FFmcOfSYZQn27NNB8VSqgR73Jk9eGjYGhDMknZOy2bxwWvzSG65uVoqz5mZNDHxXcY+dN
HCZ8u53LxMT705XgBtKaUPMUWQN3OotdPB28QsYNmiDXgqyFghCPZeifNdZJrxPohLzB5IhrVFP37mIRF3kSSURTyJ7xMlXqtkX6HM7mgjpBI6YS2qU2Ujn3
XQ== 3128990996@qq.com
PS C:\WINDOWS\system32> ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi Tsaokoo! You've successfully authenticated, but GitHub does not provide shell access.
PS C:\WINDOWS\system32>
```

（4）LaTeX 基础操作与练习：

1. 文档结构与标题：

```
\documentclass[11pt,a4paper]{article}
\usepackage[UTF8]{ctex}         % 中文支持
\title{系统开发工具基础实验报告}
\author{应佳钰}
\date{\today}
\begin{document}
\maketitle
\end{document}
```

最基本的 LaTeX 文档结构，可以生成带标题的 PDF。效果如下：

```latex
\documentclass[11pt,a4paper]{article}
\usepackage[UTF8]{ctex}        % 中文支持
\title{系统开发工具基础实验报告}
\author{应佳钰}
\date{\today}
\begin{document}
\maketitle
\end{document}
```

系统开发工具基础实验报告

应佳钰

2025 年 9 月 9 日

2. 插入章节与小节：

```latex
\section{实验目的}
\subsection{实验环境}
```

使用 \section 与 \subsection 组织文档结构。

3. 插入数学公式：

```latex
爱因斯坦质能方程： $E = mc^2$
\[
\int_0^1 x^2 dx = \frac{1}{3}
\]
```

LaTeX 支持高质量的数学公式排版：

爱因斯坦质能方程：$E = mc^2$

$$\int_0^1 x^2 dx = \frac{1}{3}$$

4. 插入表格：

```latex
\begin{tabular}{|c|c|c|}
\hline
学号 & 姓名 & 成绩 \\
\hline
001 & 张三 & 95 \\
002 & 李四 & 88 \\
\hline
\end{tabular}
```

使用 tabular 环境排版表格。

| 学号 | 姓名 | 成绩 |
|---|---|---|
| 001 | 张三 | 95 |
| 002 | 李四 | 88 |

5. 插入图片：

```
\begin{figure}[H]
    \centering
    \includegraphics[width=0.5\textwidth]{example-image}
    \caption{示 意 图}
    \label{fig:result}
\end{figure}
```

\includegraphics 用于插入图片。

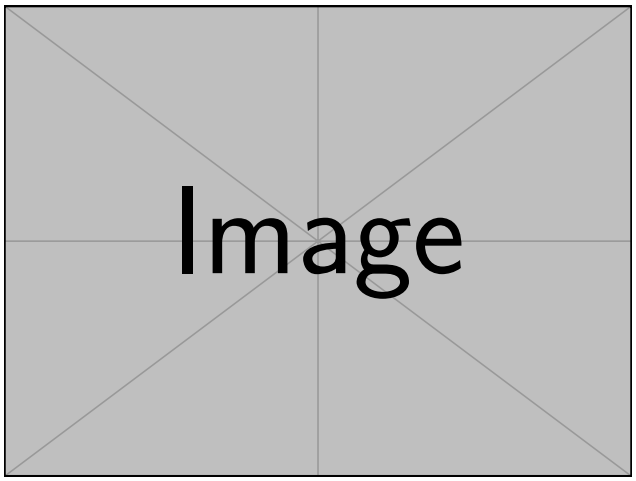



图 1: 示意图

6. 插入代码块:

```
def hello():
    print("Hello, LaTeX!")
```

借助 listings 宏包插入源代码。

```
    \begin{lstlisting}[language=python]
def hello():
    print("Hello, LaTeX!")
    \end{lstlisting}
```

7. 插入参考文献:

```
\begin{thebibliography}{9}
\bibitem{latex} LaTeX Project, \url{https://www.latex-project.org/}
\end{thebibliography}
```

提供参考文献条目。效果如下:

## 参考文献

[1] LaTeX Project, https://www.latex-project.org/

8. 使用列表：

```
\begin{itemize}
    \item 第一项
    \item 第二项
\end{itemize}
```

使用 itemize 生成无序列表。

- 第一项
- 第二项

9. 插入超链接：

```
这是 \href{https://www.github.com}{GitHub} 链接。
```

hyperref 宏包支持超链接。

这是 GitHub 链接。

10. 插入脚注：

```
这是一个脚注示例\footnote{这是脚注内容}。
```

脚注内容显示在页面底部。

这是一个脚注示例[1]。

11. 使用公式编号：

```
\begin{equation}
a^2 + b^2 = c^2
\end{equation}
```

equation 环境自动编号公式。

$$a^2 + b^2 = c^2 \tag{1}$$

12. 设置页面边距：

```
\usepackage{geometry}
\geometry{left=2.5cm,right=2.5cm,top=2.5cm,bottom=2.5cm}
```

geometry 宏包调整页面布局。

# 5 实验结果与分析

## 5.1 实验结果

本次实验成功完成了以下内容：

[1]这是脚注内容

- Git 本地仓库初始化、文件添加、提交、推送等操作均能正常执行，并在 GitHub 仓库中查看到完整的提交记录。
- 成功创建和合并功能分支，验证了 Git 分支的隔离性与合并效果，最终结果与预期一致。
- 配置 SSH 密钥并完成 GitHub 的免密连接，能够直接通过 `git push` 与远程仓库交互。
- 使用 LaTeX 编写文档，成功实现了文档结构的组织（标题、章节、小节）、数学公式、表格、图片、代码、脚注和超链接等功能。

### 5.2 结果分析

- 从 Git 部分实验可以看出，版本控制系统能够有效管理代码演变过程，通过提交历史和分支操作，既能保证开发独立性，又能方便协作。
- Git 的 SSH 免密登录避免了多次输入密码，提高了工作效率，是后续项目开发中的必要技能。
- 从 LaTeX 实例实验结果来看，LaTeX 在文档编写、公式与表格排版、参考文献管理方面具有显著优势，特别适合学术报告与科研论文写作。
- 通过在报告中插入图片和代码块，可以使实验结果更加直观，保证了实验记录的完整性与可读性。

## 6 实验总结

### 6.1 遇到的问题与解决方法

- 问题 1：用 HTTPS 推送到 GitHub 时，因网络原因经常超时。
- 解决方法：通过生成 SSH 密钥并配置到 GitHub，实现了免密推送，解决了网络不稳定带来的问题。
- 问题 2：LaTeX 表格排版时出现列对齐不整齐的问题。
- 解决方法：通过 `tabular` 环境和 `booktabs` 宏包调整表格格式，使其更加美观。

### 6.2 心得体会

通过本次实验，我学到了 Git 是现代软件开发必不可少的工具，掌握其分支与远程操作可以显著提升团队协作效率。在进行 SSH 配置时虽然步骤较多，但能够解决网络访问受限和频繁输入密码的问题，为开发环境的长期使用奠定了基础。同时 LaTeX 拥有强大的排版能力，特别适合科研文档与学术论文的撰写。相比 Word，LaTeX 在公式、参考文献和整体排版的可控性上有天然优势。

通过 Git 与 LaTeX 的结合，不仅提升了我在代码管理方面的能力，也提升了文档撰写的专业性，这对于后续课程学习和科研写作都有积极意义。

# 7 参考文献

1. Git 官方文档. https://git-scm.com/doc

2. LaTeX 项目. https://www.latex-project.org/

# 附录：GitHub 仓库链接

https://github.com/Tsaokoo/my-ouc-first-project