



中国海洋大学

系统开发工具基础实验报告

上课时间:	周五 1-4 节
姓 名:	张仕达
学 号:	23010022094
指导老师:	周小伟

1 实验内容

1.1 主题 1: git bash 的基本配置: 设置和查看用户的配置信息。

1.1.1 内容

1. 打开 Git Bash

2. 设置用户信息

```
git config --global user.name "name"
```

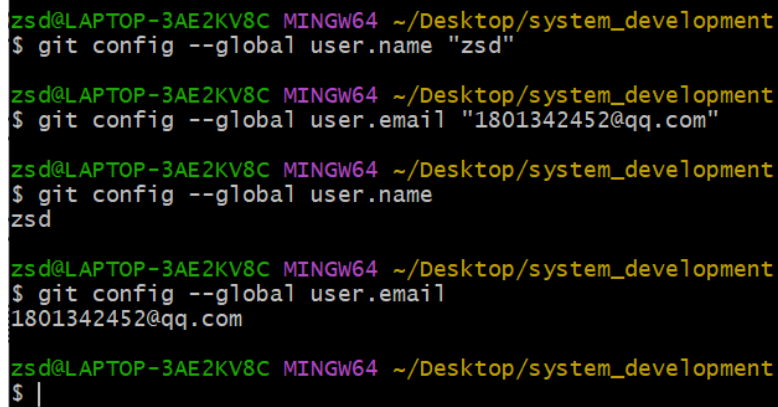
```
git config --global user.email "email"
```

3. 查看配置信息

```
git config --global user.name
```

```
git config --global user.email
```

1.1.2 结果



```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development
$ git config --global user.name "zsd"

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development
$ git config --global user.email "1801342452@qq.com"

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development
$ git config --global user.name
zsd

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development
$ git config --global user.email
1801342452@qq.com

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development
$ |
```

1.2 主题 2: git 的基本指令: 创建本地仓库。

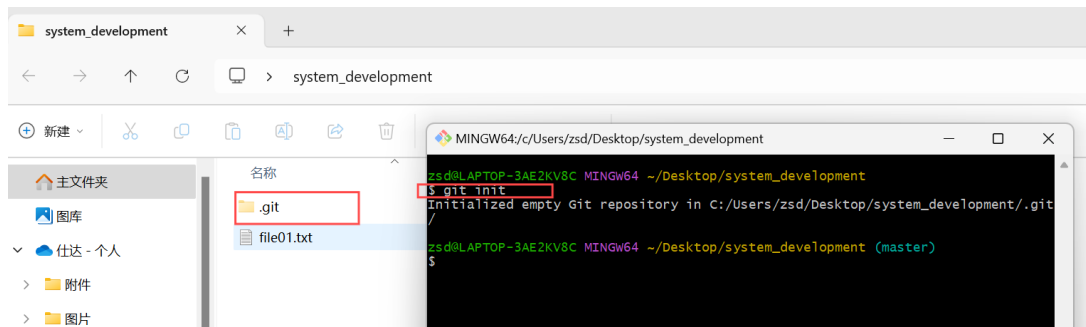
1.2.1 内容

1. 创建一个目录

2. 在目录内打开 Git Bash 窗口, 执行命令 `git init`, 将当前目录初始化为一个仓库

3. 创建成功后生成一个隐藏文件 `.git`

1.2.2 结果

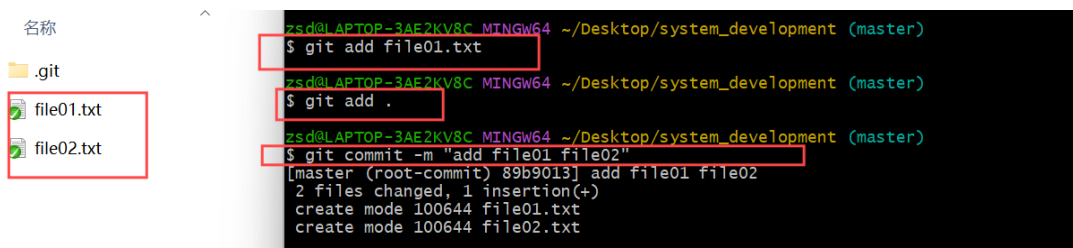


1.3 主题 3: git 的操作指令: 提交文件至本地仓库。

1.3.1 内容

1. 将当前目录内的文件提交到暂存区 `git add filename`
2. 将所有文件都提交到暂存区 `git add .`
3. 将暂存区的文件提交到本地仓库
`git commit -m '注释内容'`

1.3.2 结果

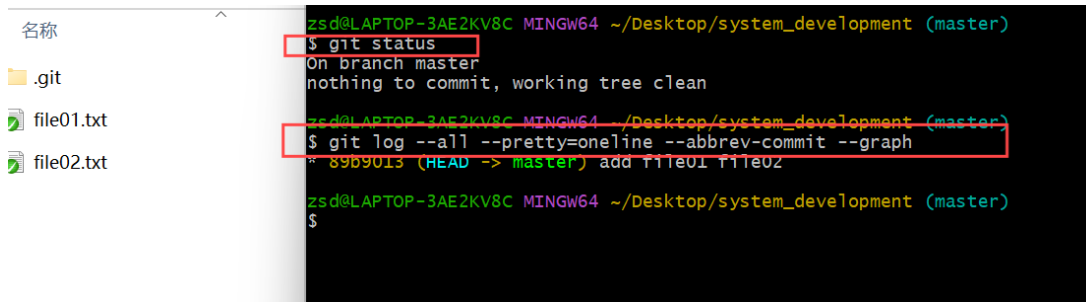


1.4 主题 4: git 的操作指令: 查看文件状态、查看提交日志

1.4.1 内容

1. 每次提交文件到暂存区或本地仓库都可用 `git status` 查看文件状态
2. 查看提交日志 `git log [option]`
其中 [option]:
-all 显示所有分支
-pretty=oneline 将提交信息显示为一行
-abbrev-commit 使得输出的 commitId 更简短
-graph 以图的形式显示

1.4.2 结果



```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git status
On branch master
nothing to commit, working tree clean

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git log --all --pretty=oneline --abbrev-commit --graph
* 89b9013 (HEAD -> master) add file01 file02

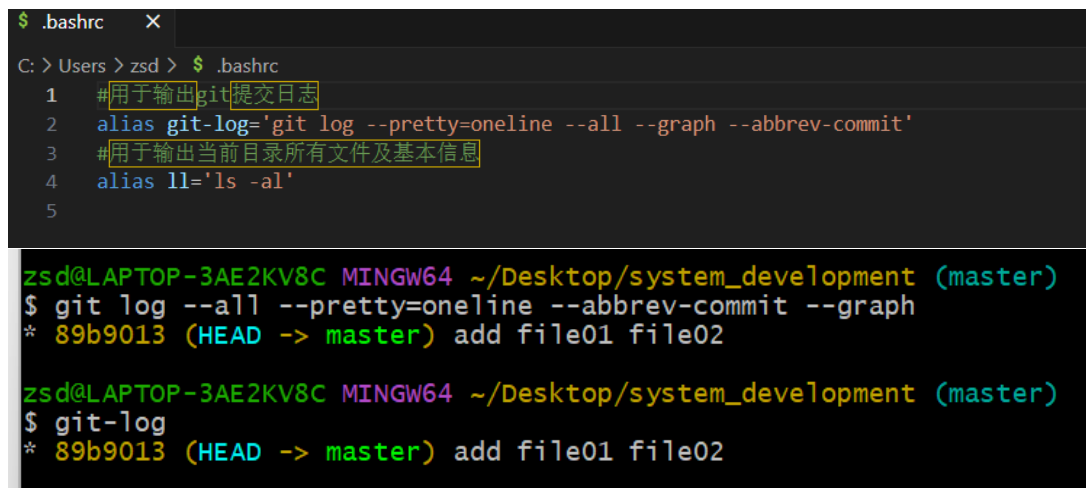
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$
```

1.5 主题 5: git 的操作指令: 为常用指令配置别名

1.5.1 内容

1. 打开用户目录, 创建.bashrc 文件, 输入如下内容 (举例):
用于输出 git 提交日志:
alias git-log='git log --pretty=oneline --all --graph --abbrev-commit'
用于输出当前目录所有文件及基本信息
alias ll='ls -al':
2."=" 右边" 内的指令可用左边指令代替, 使用起来更加简便

1.5.2 结果



```
$ .bashrc
C: > Users > zsd > $ .bashrc
1 #用于输出git提交日志
2 alias git-log='git log --pretty=oneline --all --graph --abbrev-commit'
3 #用于输出当前目录所有文件及基本信息
4 alias ll='ls -al'
5

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git log --all --pretty=oneline --abbrev-commit --graph
* 89b9013 (HEAD -> master) add file01 file02

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git-log
* 89b9013 (HEAD -> master) add file01 file02
```

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ ls -al
total 21
drwxr-xr-x 1 zsd 197121 0 Sep  5 23:24 ./
drwxr-xr-x 1 zsd 197121 0 Sep  5 22:54 ../
drwxr-xr-x 1 zsd 197121 0 Sep  5 23:28 .git/
-rw-r--r-- 1 zsd 197121 7 Sep  5 23:00 file01.txt
-rw-r--r-- 1 zsd 197121 0 Sep  5 23:24 file02.txt

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ ll
total 21
drwxr-xr-x 1 zsd 197121 0 Sep  5 23:24 ./
drwxr-xr-x 1 zsd 197121 0 Sep  5 22:54 ../
drwxr-xr-x 1 zsd 197121 0 Sep  5 23:28 .git/
-rw-r--r-- 1 zsd 197121 7 Sep  5 23:00 file01.txt
-rw-r--r-- 1 zsd 197121 0 Sep  5 23:24 file02.txt
```

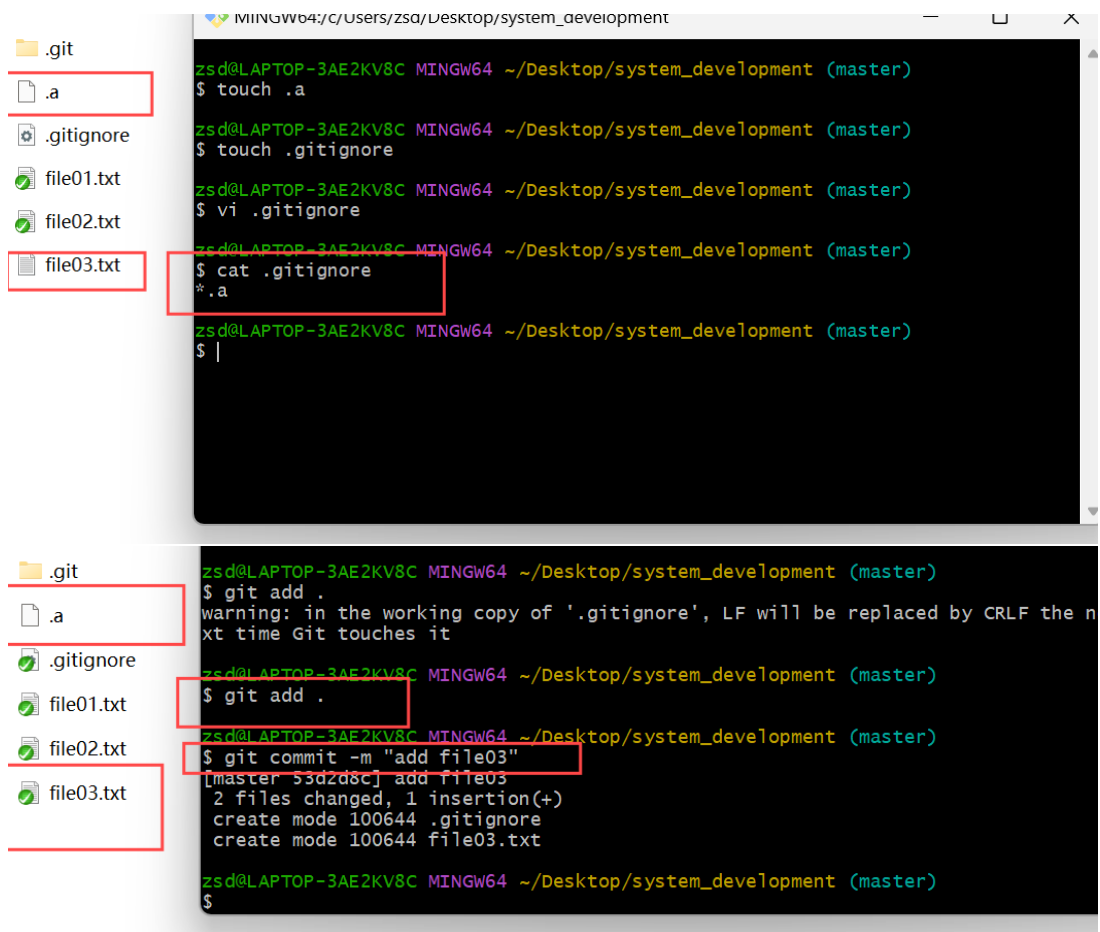
1.6 主题 6: git 的操作指令: 添加文件至忽略列表

1.6.1 内容

1. 在当前目录中创建一个名为.gitignore 的文件
2. 在目录中输入相关信息可使相关文件不受 git 管理

例: 输入 *.a 表示忽略目录中所有以.a 结尾的文件

1.6.2 结果



The image shows two screenshots of a terminal window with a file explorer on the left. The file explorer lists: .git, .a, .gitignore, file01.txt, file02.txt, and file03.txt. The terminal window title is 'MINGW64: c:/Users/zsa/Desktop/system_development'.

Top Screenshot:

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ touch .a
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ touch .gitignore
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ vi .gitignore
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ cat .gitignore
*.a
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ |
```

Bottom Screenshot:

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the
next time Git touches it
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git add .
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git commit -m "add file03"
[master 53d2d8c] add file03
2 files changed, 1 insertion(+)
create mode 100644 .gitignore
create mode 100644 file03.txt
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$
```

1.7 主题 7: git 的操作指令: 创建分支、切换分支

1.7.1 内容

1. 创建分支 `git branch 分支名`
2. 切换分支 `git checkout 分支名`
3. 查看当前分支 `git branch`

1.7.2 结果

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git branch dev
HEAD -> master, dev
add file03
add file01 file02

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git checkout dev
Switched to branch 'dev'
M file02.txt

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (dev)
$ git-log
* 53d2d8c (HEAD -> dev, master) add file03
* 89b9013 add file01 file02

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git checkout dev
Switched to branch 'dev'

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (dev)
$ git branch
* dev
master
```

当前位于master

已切换到dev分支

1.8 主题 8: git 的操作指令: 解决合并冲突、合并分支、删除分支

1.8.1 内容

1. 将一个分支上的提交合并到另一个分支 `git merge 分支名`
2. 如果两个分支对文件的修改存在冲突则无法合并
3. 需要对被合并的文件进行修改后再合并
4. `git branch -d b1` 删除分支

1.8.2 结果

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (dev)
$ git checkout master
Switched to branch 'master'

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git merge dev
Auto-merging file02.txt
CONFLICT (content): Merge conflict in file02.txt
Automatic merge failed; fix conflicts and then commit the result.

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master|MERGING)
$ cat file02.txt
<<<<<< HEAD
matser
=====
dev
>>>>>> dev
```

master 中的内容
dev 中的内容
内容冲突所以失败了

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git merge dev
Already up to date.

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ cat file02.txt
master
```

将dev中的内容删去, 再次合并
内容变成master中的内容

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git-log
* a96726a (HEAD -> master) change file02 master
* e0f99d2 change file02
| \
| * cc5a884 (dev) change file02 dev
| * | 5a9de6a change file02 master
| /
* 53d2d8c add file03
* 89b9013 add file01 file02
```

此处合并到一起
此处产生分支

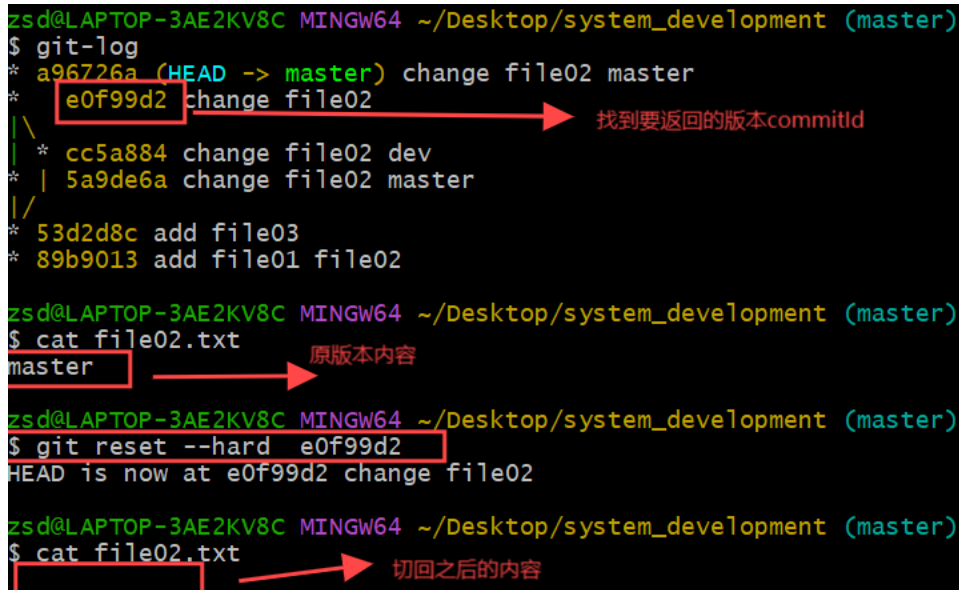
1.9 主题 9: git 的操作指令: 版本回退

1.9.1 内容

git reset --hard commitID

commitID 可以使用 git-log 或 git log 指令查看

1.9.2 结果



```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git-log
* a96726a (HEAD -> master) change file02 master
* e0f99d2 change file02
| \
| * cc5a884 change file02 dev
* | 5a9de6a change file02 master
| /
* 53d2d8c add file03
* 89b9013 add file01 file02

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ cat file02.txt
master

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git reset --hard e0f99d2
HEAD is now at e0f99d2 change file02

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ cat file02.txt
```

找到要返回的版本commitId

原版本内容

切回之后的内容

1.10 主题 10: git 的操作指令: 添加远程仓库

1.10.1 内容

1. 初始化本地库之后, 与已创建的远程仓库进行对接 (示例为 github)

命令: git remote add < 远端名称 > < 仓库路径 >

远端名称, 默认是 origin, 取决于远端服务器设置

2. 查看远端仓库 git remote

1.10.2 结果

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git remote add origin git@github.com:coff-sug/System-development-tool.git

zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git remote
origin
```

1.11 主题 11: git 的操作指令: 将文件推送到远程仓库

1.11.1 内容

命令: `git push [远端名称] [本地分支名]: [远端分支名]`

如果远程分支名和本地分支名称相同, 则可以只写本地分支

`git push origin master`

如果当前分支已经和远端分支关联, 则可以省略分支名和远端名。

`git push` 将 `master` 分支推送到已关联的远端分支

1.11.2 结果

```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/system_development (master)
$ git push origin master
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (16/16), 1.32 KiB | 676.00 KiB/s, done.
Total 16 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:coff-sug/System-development-tool.git
 * [new branch]      master -> master
```

master	1 Branch	0 Tags	Go to file	Add file	Code
coff-sug change file02 e0f99d2 · 52 minutes ago 5 Commits					
.gitignore	add file03	1 hour ago			
file01.txt	add file01 file02	3 days ago			
file02.txt	change file02	52 minutes ago			
file03.txt	add file03	1 hour ago			

1.12 主题 12: git 的操作指令: 从远程仓库克隆

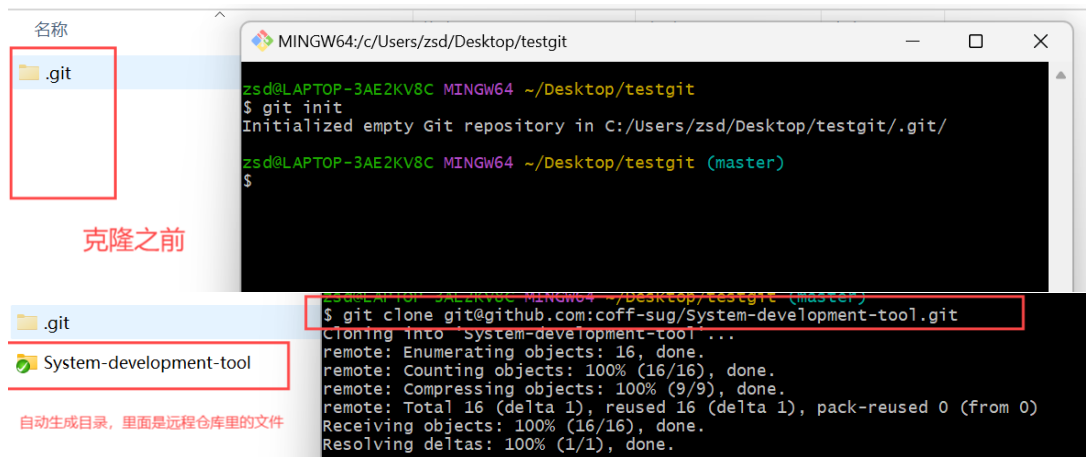
1.12.1 内容

如果已经有一个远端仓库, 我们可以直接 clone 到本地。

命令: `git clone < 仓库路径 > [本地目录]`

本地目录可以省略, 会自动生成一个目录

1.12.2 结果



1.13 主题 13: git 的操作指令: 从远程仓库抓取、拉取

1.13.1 内容

1. 抓取命令: `git fetch [remote name] [branch name]`

抓取指令就是将仓库里的更新都抓取到本地, 不会进行合并

如果不指定远端名称和分支名, 则抓取所有分支。

2. 拉取命令: `git pull [remote name] [branch name]`

拉取指令就是将远端仓库的修改拉到本地并自动进行合并, 等同于 `fetch+merge`

如果不指定远端名称和分支名, 则抓取所有并更新当前分支

1.13.2 结果

The image displays three sequential screenshots of a terminal window showing the results of Git operations, with a file explorer on the left showing the project structure.

First Screenshot: The terminal shows the command `$ git fetch` being executed. The output indicates that the remote repository was updated, and the local repository was updated to match the remote. The file explorer on the left shows the project structure, including `.git`, `.gitignore`, `file01.txt`, `file02.txt`, and `file03.txt`.

Second Screenshot: The terminal shows the command `$ git merge` being executed. The output indicates that the merge was successful, and the local repository was updated to match the remote. The file explorer on the left shows the project structure, including `.git`, `.gitignore`, `file01.txt`, `file02.txt`, `file03.txt`, and `file04.txt`.

Third Screenshot: The terminal shows the command `$ git pull` being executed. The output indicates that the pull was successful, and the local repository was updated to match the remote. The file explorer on the left shows the project structure, including `.git`, `.gitignore`, `file01.txt`, `file02.txt`, `file03.txt`, `file04.txt`, and `file05.txt`.

1.14 主题 14: latex 对标题、作者、时间和引用的编辑

1.14.1 内容

在导言区插入标题、作者、时间的信息，在正文区进行引用
在标题之后加入引用

1.14.2 结果

```
\title{标题} % 文章标题
\author{作者名} % 作者的名称
\date{\today} % 当天日期
```

```
\begin{document}
```

```
\maketitle %引入
```

```
\begin{abstract}
```

该部分内容是放置摘要信息的。

```
\end{abstract}
```

标题

作者名

2024 年 9 月 10 日

摘要

该部分内容是放置摘要信息的。

1.15 主题 15: latex 的标题架构

1.15.1 内容

标题设置: 一级标题 section, 二级标题 subsection, 三级标题 subsubsection;

段落设置: 在一段的最后添加 par 代表一段的结束:

目录设置: 在 begindocument 内容中添加 tableofcontents

```
\begin{document}|

% 生成目录设置
\renewcommand{\contentsname}{目录} %将content转为目录
\tableofcontents

% 标题开始
\section{一级标题1}
第一段一级标题下的内容, 一级标题下的内容, 一级标题下的内容, 一级标题下的内容。 \par
第二段一级标题下的内容, 一级标题下的内容, 一级标题下的内容, 一级标题下的内容。

\subsection{二级标题1.1}
二级标题下的内容。

\subsubsection{三级标题下的内容1.1.1}
三级标题下的内容。

\section{一级标题2}
一级标题2中的内容
```

1.15.2 结果

目录

1 一级标题 1 1

1.1 二级标题 1.1 1

1.1.1 三级标题下的内容 1.1.1 1

2 一级标题 2 1

3 系统概述 3

1 一级标题 1

第一段一级标题下的内容，一级标题下的内容，一级标题下的内容，一级标题下的内容。

第二段一级标题下的内容，一级标题下的内容，一级标题下的内容，一级标题下的内容。

1.1 二级标题 1.1

二级标题下的内容。

1.1.1 三级标题下的内容 1.1.1

三级标题下的内容。

2 一级标题 2

一级标题 2 中的内容

1.16 主题 16: latex 插入图片

1.16.1 内容

需要导入宏包：usepackagegraphicx

导入图片所在的路径：graphicspathfigures/

正文区引入图片：includegraphics[]logo

1.16.2 结果

```
\graphicspath{{figures/}}%图片位于figures文件夹内
\usepackage{graphicx}

\begin{document}

\includegraphics[width=0.8\textwidth]{logo}\\%[]内设置图片参数 {}内输入图片名称
```



中国海洋大学

1.17 主题 17: latex 插入表格

1.17.1 内容

```
\begin{table}[htbp] % htp代表表格浮动位置
% 表格居中
\centering
% 添加表头
\caption{变量表}
% 创建table环境
\begin{tabular}{|cc|c|} % 3个c代表3列都居中，也可以设置l或r，|代表竖线位置
% 表格的输入
\hline % 一条水平线
x & y & z \\ % \\为换行符
\hline
11 & 22 & 33 \\
\hline
\end{tabular}
\end{table}
```

1.17.2 结果

```
\begin{table}[htbp] % htp代表表格浮动位置
% 表格居中
\centering
% 添加表头
\caption{变量表}
% 创建table环境
\begin{tabular}{|cc|c|} % 3个c代表3列都居中，也可以设置l或r，|代表竖线位置
% 表格的输入
\hline % 一条水平线
x & y & z \\ % \\为换行符
\hline
11 & 22 & 33 \\
\hline
\end{tabular}
\end{table}
```

1.18 主题 18: latex 改变字体、大小

1.18.1 内容

使用代码：字体内容

使用代码：大小内容

1.18.2 结果

```
\begin{document}  
{\songti 宋体}  
{\heiti 黑体}  
{\fangsong 仿宋}  
{\kaishu 楷书}
```

```
{\bf 粗体}  
{\it 斜体}  
{\sl 斜体}
```

```
\textbf{粗体}  
\textit{斜体}|  
\textsl{斜体}
```

```
{\tiny Hello} \\  
{\scriptsize Hello} \\  
{\footnotesize Hello} \\  
{\small Hello} \\  
{\normalsize Hello} \\  
{\large Hello} \\
```

宋体 黑体 仿宋 楷书

粗体 斜体 斜体

粗体 斜体 斜体¹⁸

Hello

Hello

1.19 主题 19: latex: 使用算法（伪代码）

1.19.1 内容

```
\documentclass{article}
\usepackage[UTF8]{ctex}
\usepackage{algorithm} % 排版算法
\usepackage{algorithmic} % 排版算法

\title{Algorithm}
\author{NSJim Green}
\date{October 2020}

\begin{document}

\maketitleS

\section{Algorithm 1}

\begin{algorithm}
\caption{Checksum(A, x)} %算法标题
\label{alg2} %标签
\begin{algorithmic} %算法开始
\STATE {\bf Input:} An array A and a value x %也可以用\textbf{Input:}
\STATE {\bf Output:} A bool value show if there is two elements in A whose sum is x
\STATE A $\gets$ SORT(A)
\STATE n $\gets$ length(n)
\FOR i $\gets$ 0 to n
  \IF Binary-search(A, x-A[i], 1, n)
    \STATE return true
  \ENDIF
\ENDFOR
\STATE return false
\end{algorithmic}
\end{algorithm}

\end{document}
```

1.19.2 结果

1 Algorithm 1

Algorithm 1 CheckSum(A,x)

Input: An array A and a value x

Output: A bool value show if there is two elements in A whose sum is x

A \leftarrow SORT(A)

n \leftarrow length(n)

for i \leftarrow 0 to n **do**

if Binary-search(A,x-A[i],1,n) **then**

 return true

end if

end for

return false

1.20 主题 20: latex: 使用代码块

1.20.1 内容

```
\documentclass{article}
\usepackage[UTF8]{ctex}
\usepackage{listings}

% 代码块基础设置
\lstset{
  numbers=left,           % 在左侧显示行号
  showstringspaces=false, % 不显示字符串中的空格
  frame=single,          % 设置代码块边框
}

\title{Code block}
\author{NSJim Green}
\date{October 2020}

\begin{document}

\maketitle

\section{C Language}

\begin{lstlisting}[language=c]
#include <stdio.h>

// main function
int main() {
    printf("Hello World!");
    return 0;
}
\end{lstlisting}

\end{document}
```

1.20.2 结果

1 C Language

```
1 #include <stdio.h>
2
3 // main function
4 int main() {
5     printf("Hello World!");
6     return 0;
7 }
```

2 解题感悟

git 和 latex 的学习对我来说是非常有益的。git 让我实现了在本地仓库与远程仓库上对代码的管理，我可以使用 git 和其他人方便地共同开发一个项目，也可以将我的代码成果保存起来供其他人评价，更重要的是供我以后的学习回顾。而学习 latex 则教会了我对报告、论文模版的编辑，这对我以后进行数学建模等比赛有很大的裨益。

[github 地址 \[点击\]](#)