

```
guoyi@DESKTOP-GKKHCEA MINGW64 ~/D:transferred_files
(main)
$ git clone C:\Users\86156\Documents
Cloning into 'Users86156Documents'...
warning: You appear to have cloned an empty repository.
done.什么意思
```

这段输出是你执行 `git clone C:\Users\86156\Documents` 命令后 Git 给出的反馈，下面为你详细解释其含义：

#### 命令执行情况

```
bash ^
git clone C:\Users\86156\Documents
```

此命令的目的是将 `C:\Users\86156\Documents` 这个路径下的 Git 仓库克隆到当前目录 (`D:transferred_files`)。

#### 输出信息分析

1. Cloning into 'Users86156Documents'...

🖼️ 图像生成 📄 AI 编程 📝 帮我写作 🔍 AI 搜索 📖 AI 阅读 🔍 学术搜索 📝 解题答疑 📁 更多

尝试把 C 盘的东西转到 D 盘，但是修改不了，有红色的地方，用豆包给的代码也改变不过去，但是在 C 盘建立了新的文件夹，但文件夹里面没有克隆进去的文件

/ 的含义

在类 Unix 系统（如 Linux、macOS）以及 Windows 系统的 Git Bash 环境中，`/` 是路径分隔符，用于分隔路径中的不同目录层级。例如，`C:/Users/username/Documents` 表示从根目录开始，依次经过 `C:` 磁盘、`Users` 文件夹、`username` 文件夹，最终到达 `Documents` 文件夹。在输出中，文件夹后面的 `/` 通常用于标识这是一个文件夹，而不是文件。

```
guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
D:/Git          401945596 108227104 293718492  27% /
C:              83886076  67404444  16481632  81% /c

guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
D:/Git          384G  104G   281G   27% /
C:              80G   65G   16G   81% /c

guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$
```

C 盘占用的太多了

```
guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
D:/Git          384G  104G   281G   27% /
C:              80G   65G   16G    81% /c

guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ du
0      ./anaconda
5      ./android
100    ./AndrewsBox
1048   ./BigNoxA
4      ./BigNoxR7
2      ./conda
1      ./continuum/anaconda-client
1      ./continuum
53     ./git/hooks
2      ./git/info
1      ./git/logs/refs/heads
1      ./git/logs/refs
2      ./git/logs
145    ./git/objects/00
156    ./git/objects/01
127    ./git/objects/02
139    ./git/objects/03
412    ./git/objects/04
154    ./git/objects/05
104    ./git/objects/06
101    ./git/objects/07
409    ./git/objects/08
```

Ctrl+c 是打断，du 扫描文件，会很多

`du -s * | sort -nr > ~/report.txt` 是一条在类 Unix 系统（如 Linux、macOS）或者 Windows 的 Git Bash 环境下使用的命令组合，下面为你详细解释它的含义和各部分功能。

命令拆分解释

1. `du -s *`

`du` 是“disk usage”的缩写，主要用于估算文件或目录的磁盘使用空间。

`-s` 是 `du` 命令的一个选项，它的作用是仅显示每个参数的总计大小，也就是只输出每个目录的总大小，而不会详细列出该目录下子目录和文件的大小。

`*` 是通配符，表示当前目录下的所有文件和文件夹。

综合起来，`du -s *` 会计算当前目录下每个文件和文件夹所占用磁盘空间的大小，并输出每个文件或文件夹的总计大小。

2. |

这是管道符号，它的作用是将前一个命令（`du -s *`）的输出作为后一个命令（`sort -nr`）的输入。

3. `sort -nr`

`sort` 命令用于对文本行进行排序。

`-n` 选项表示按照数值大小进行排序，而不是按照字母顺序排序。

`-r` 选项表示以倒序（从大到小）的方式进行排序。

因此，`sort -nr` 会将 `du -s *` 输出的结果按照磁盘使用空间大小从大到小进行排序。

4. `> ~/report.txt`

`>` 是重定向符号，它的作用是将前面命令（`sort -nr`）的输出结果重定向到指定的文件中。

`~/` 表示用户的主目录，例如在 Linux 系统中，普通用户的主目录通常是 `/home/username`，其中 `username` 是用户的用户名。

`report.txt` 是指定的文件名。

综合起来，`> ~/report.txt` 会把 `sort -nr` 排序后的结果保存到用户主目录下的 `report.txt` 文件中。

总结

`du -s * | sort -nr > ~/report.txt` 这条命令的整体功能是：计算当前目录下所有文件和文件夹的磁盘使用空间大小，然后按照大小从大到小进行排序，最后将排序结果保存到用户主目录下的 `report.txt` 文件中。这个文件可以帮助你快速了解当前目录下各个文件和文件夹的磁盘占用情况，找出占用空间较大的文件或文件夹。

←

↺

🔒

https://gitcode.com/create?position=nav\_top

≡

GitCode

🔍 按下 / 快速开启搜索

新建项目

需要导入已有项目?

使用模板

你可以选择一个模板项目来快速新建项目

▼

仓库类型

☒ 代码仓

☐ 模型仓

☐ 数据集仓

☐ Space

项目名称

mywork

项目路径

Sev\_Snape

▼

/

mywork

项目描述

个人信息和代码

☐ 添加初始化 README 文件

(包括建议的项目的描述、安装指南、使用示例、贡献指南等信息)

☐ 添加 .gitignore 文件

选择 .gitignore 模板

▼

☐ 添加 LICENSE 文件

选择 LICENSE 模板

▼

☐ 公开项目

🔗

☒ 私密项目

🔒

取消

创建项目

## 建立私立项目

### 当前项目代码仓是空仓库

你可以通过克隆仓库开始或使用以下方式为你的项目添加文件：

HTTPSSSHhttps://gitcode.com/Sev\_Snape/mywork.git

🔗

或为你的项目添加以下文件：

📄 添加 README.md

📄 添加 LICENSE

📄 添加 .gitignore

### 命令行指引

你还可以按照以下说明从你的电脑中上传现有文件或项目。

### Git 全局设置

```
git config --global user.name "Sev_Snape"
git config --global user.email "Sev_Snape@noreply.gitcode.com"
```

### 创建一个新仓库

```
git clone https://gitcode.com/Sev_Snape/mywork.git
cd mywork
```

```
echo "# mywork" > README.md
```

## 创建一个新仓库

```
git clone https://gitcode.com/Sev_Snape/mywork.git
cd mywork
echo "# mywork" >> README.md
git add README.md
git commit -m "add README"
git branch -m main
git push -u origin main
```

## 推送现有的文件

```
cd existing_folder
git init
git remote add origin https://gitcode.com/Sev_Snape/mywork.git
git add .
git commit -m "Initial commit"
git branch -m main
git push -u origin main
```

## 推送现有的 Git 仓库

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitcode.com/Sev_Snape/mywork.git
git push -u origin --all
git push -u origin --tags
```

[https://gitcode.com/Sev\\_Snape/mywork.git](https://gitcode.com/Sev_Snape/mywork.git)

git@gitcode.com:Sev\_Snape/mywork.git

```

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ mkdir repo

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ cd repo

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo (main)
$ pwd
/c/Users/86156/repo/repo

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo (main)
$ git clone git@gitcode.com:Sev_Snape/mywork.git
Cloning into 'mywork'...
warning: You appear to have cloned an empty repository.

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo (main)
$ ls -l
total 0
drwxr-xr-x 1 guoyi 197121 0   3月 18 11:48 mywork/

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo (main)
$ cd mywork/

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo/mywork (main)
$ ls -l
total 0

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo/repo/mywork (main)
$

```

```

MINGW64:/c/Users/86156/repo  X  +  v  -  □  X

remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), 8.45 KiB | 4.22 MiB/s, done.

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ git clone git@gitcode.com:Sev_Snape/week02.git
fatal: destination path 'week02' already exists and is not an empty directory.

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ ^C

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ cd

guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ pwd
/c/Users/86156

guoyi@DESKTOP-GKKHCEA MINGW64 ~ (main)
$ cd repo

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$ git clone git@gitcode.com:Sev_Snape/week02.git
fatal: destination path 'week02' already exists and is not an empty directory.

guoyi@DESKTOP-GKKHCEA MINGW64 ~/repo (main)
$

```

不知道为什么这样打开不了 week02 了，但在下面能找到 ↓



The screenshot shows a VS Code editor window with a dark theme. The Explorer sidebar on the left shows a file tree with 'WEEK01' selected. The main editor area displays a 'README.md' file with the following content:

```
1 # 提交 PDF 学习报告
2
3 ## 操作步骤
4
5 首先，将本课程仓库（你只能查看，不能修改）Fork 至你个人名下的同名仓库（你有修改权限），不必创建新分支。
6
7 然后，点击你个人仓库“代码”页面右上方的`Clone`按钮，复制你个人名下同名仓库的SSH地址（注意，不是`https://gitcode.com`开头的地址，而是`git@gitcode.com`开头的地址；前者只能下载，不能上推），在终端（Windows 可以用安装的 Git Bash）里，先用`cd`命令切换至你准备存放本地代码仓库的上级目录，然后用`git clone`命令把仓库克隆到本地。
8
9 如果能够在本地计算机文件夹里打开看到此文件的内容（推荐用 VS Code 打开你的本地仓库文件夹），就说明你成功地把本仓库 Clone 到本地了。
10
11 接下来，你可以把本周学习过程中记录产生的 PDF 报告复制或移动到仓库文件夹里来，然后在终端（Git Bash）里运行以下命令（需要适当指定参数）：
12
13 - 用`git add`命令将 PDF 报告加入版本管理
14 - 用`git commit`命令将代码改动（其实就是添加的 PDF 文件）确认提交到本地版本历史中
15 - 用`git push`命令将本地新提交的版本历史上推到 GitCode 平台你个人名下的同名仓库中
16
```

左边没有，右边有