

## Week2 学习笔记

一些命令: .....	2
任务一: 学习 ls 命令 .....	3
基础知识 .....	3
命令行与文件系统 .....	3
根目录 .....	5
路径,绝对路径,相对路径 .....	6
理解 Shell (Bash、Zsh) 的基本语法结构 (空格分隔、短选项、长选项、参数)与命令 ls .....	6
任务二,三: .....	12
任务三 .....	12
命令 cp .....	15
命令 mv .....	16
命令 mkdir .....	20
命令 rm .....	22
命令 df .....	24
命令 du .....	26
任务四:用 df 查看磁盘剩余空间, 使用 du 命令查看文件/文件夹占用的磁盘空间 .....	27
任务五: .....	29
任务六: .....	30

## 一些命令：

`touch 文件名.py` #在当前目录创建文件（文件名.py）

`code 文件名.py` #在 vs code 打开文件（文件名.py）

`cd /d` #将路径切换为 d 盘

路径空格处理：如果路径包含空格，需要用引号包裹路径

如：`cd "/d/My Projects/Important Files"`

# 任务一：学习 ls 命令

## 1. 路径格式：

- Git Bash 使用 Unix 风格的路径，盘符（如 D 盘）对应为 `/d/`。
- **斜杠方向**：必须使用 `/`（正斜杠），而非 Windows 默认的 `\`（反斜杠）。

## 基础知识

### 命令行与文件系统

1. 学习使用 ls 命令，检查自己计算机最常用的“桌面”、“下载”、“文档”等文件夹的真实的文件系统路径是什么

```
ymq@LAPTOP-J3NTE080 MINGW64 /c
$ pwd
/c

ymq@LAPTOP-J3NTE080 MINGW64 /c
$ ls
'$Recycle.Bin'/          PerfLogs/
AMFTTrace.log           'Program Files'/
CloudMusic/             'Program Files (x86)'/
Config.Msi/             ProgramData/
'Documents and Settings'@ Recovery/
DumpStack.log           swapfile.sys
DumpStack.log.tmp       'System Volume Information'/
hiberfil.sys            Users/
msdia80.dll*            WeGameApps/
NVIDIA/                 Windows/
pagefile.sys            XmpCache/
```

命令行输入时可以用 tab 补全,连接两次 tab 可以显示多个补全结果

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ cd WPS
WPS Cloud Files/ WPSDrive/
```

桌面路径

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ pwd
/c/Users/lxm/Desktop
```

下载路径

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/Downloads
$ pwd
/c/Users/lxm/Downloads
```

文档路径

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ cd My\ Documents/

ymq@LAPTOP-J3NTE080 MINGW64 ~/My Documents
$ pwd
/c/Users/lxm/My Documents
```

用 ls -a 显示以.开头的文件

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ ls -a
./
../
.anaconda/
.conda/
.condarc
.continuum/
.douyu_channel/
.gitconfig
.idlrc/
.ipython/
.lessht
.ssh/
.vscode/
.xp2p/
「开始」菜单@
ado/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
Links/
Local Settings@
```

对比 ls,不显示.开头文件

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ ls
「开始」菜单@
ado/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
Links/
'Local Settings'@
Music/
'My Documents'@
NetHood@
NTUSER.DAT
ntuser.dat.LOG1
ntuser.dat.LOG2
NTUSER.DAT{357b2189-f6e4-11ee-9ef5-b946b238e1fe}.TM.blf
NTUSER.DAT{357b2189-f6e4-11ee-9ef5-b946b238e1fe}.TMContainer00000000000000000001.regtrans-ms
NTUSER.DAT{357b2189-f6e4-11ee-9ef5-b946b238e1fe}.TMContainer00000000000000000002.regtrans-ms
```

将目录改为桌面,用 ls 查看所有文件,用 cat 显示某一文件中的内容

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ cd Desktop/

ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ ls
1yan.lnk*                  瓦配置.txt
1研.lnk*                  微信截图_20240807013607.png
desktop.ini                微信截图_20240807013629.png
'Microsoft Edge.lnk'*     微信截图_20240807013716.png
' '$\360\237\232\227' '问题汇总.txt' 微信截图_20240903103300.png
百度网盘同步空间.lnk*    微信截图_20240906121831.png
报告上传问题.docx        应用/
崩铁+原.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ cat 瓦配置.txt
准星:
0;s;1;P;c;7;o;0.213;d;1;z;3;0b;0;1b;0;S;s;1.007;o;0.475
```

## 根目录

根目录就是斜杠"/", Windows 从盘符开始, Unix 从/开始  
cd 到主目录, cd /到根目录

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ cd /

ymq@LAPTOP-J3NTE080 MINGW64 /
$ cd
```

## 路径,绝对路径,相对路径

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ cat 瓦配置.txt
```

其中'瓦配置.txt',就是**相对路径**的写法,相对于上面 desktop.只要不是从根目录开始的就是相对路径,相对的时当前工作目录,比如在下图中当前工作目录为~,那么 cat desktop/瓦配置.txt 命令中下划线部分也属于相对路径,

```
ymq@LAPTOP-J3NTE080 MINGW64 ~
$ cat /c/Users/lxm/Desktop/瓦配置.txt
准星:
```

上面就是从 '/' 也就是根目录开始的**绝对路径**写法(在写绝对路径时也可以用 tab 进行补全)

".."为上一级文件夹,"."当前文件夹

## 理解 Shell (Bash、Zsh) 的基本语法结构 (空格分隔、短选项、长选项、参数)与命令 ls

命令行也是一个程序

ls Desktop/,中空格后的 desktop/就是参数,

### SYNOPSIS [top](#)

```
ls [OPTION]... [FILE]...
```

### DESCRIPTION [top](#)

List information about the FILES (the current directory by default). Sort entries alphabetically if none of **-cftuvSUX** nor **--sort** is specified.

Mandatory arguments to long options are mandatory for short options too.

**-a, --all**  
do not ignore entries starting with .

图中 -a 就是短选项,--all 就是长选项,二者等价

Synopsis 下为语法结构

Zsh 可以看作 Bash 的扩展,其功能更强大

```

ymq@LAPTOP-J3NTE080 MINGW64 ~
$ ls -a
./
../
.anaconda/
.conda/
.condarc
.continuum/
.douyu_channel/
.gitconfig
.idlrc/
.ipynb/
.lesshst
.ssh/
.vscode/
.xp2p/
「开始」菜单@
ado/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/

```

不同文件类型有不同颜色,途中白色为普通文件,蓝色加斜杠为文件夹  
不同操作系统中命令行定义可能有区别

```

ymq@LAPTOP-J3NTE080 MINGW64 ~
$ ls -a --color=never
./
../
.anaconda/
.conda/
.condarc
.continuum/
.douyu_channel/
.gitconfig
.idlrc/
.ipynb/

```

ls -a --color=never #消除颜色

```

ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ ls -al
total 4664
drwxr-xr-x 1 ymq 197121 0 3月 13 14:51 ./
drwxr-xr-x 1 ymq 197121 0 3月 13 11:25 ../
-rw-r--r-- 1 ymq 197121 162 3月 13 14:51 '~$报告上传问题.docx'
-rwxr-xr-x 1 ymq 197121 976 3月 11 21:08 1yan.lnk*
-rwxr-xr-x 1 ymq 197121 947 3月 11 21:08 1研.lnk*
-rw-r--r-- 1 ymq 197121 282 5月 6 2024 desktop.ini
-rwxr-xr-x 1 ymq 197121 2379 3月 12 08:54 'Microsoft Edge.lnk'*
-rw-r--r-- 1 ymq 197121 901 5月 13 2024 '$'\360\237\232\227'问题汇总.txt'
-rwxr-xr-x 1 ymq 197121 498 3月 13 09:41 百度网盘同步空间.lnk*
-rw-r--r-- 1 ymq 197121 374093 3月 12 21:44 报告上传问题.docx
-rw-r--r-- 1 ymq 197121 559 7月 6 2024 崩铁+原.txt
-rw-r--r-- 1 ymq 197121 69 11月 14 18:36 瓦配置.txt
-rw-r--r-- 1 ymq 197121 1128983 8月 7 2024 微信截图_20240807013607.png
-rw-r--r-- 1 ymq 197121 1307626 8月 7 2024 微信截图_20240807013629.png
-rw-r--r-- 1 ymq 197121 1207682 8月 7 2024 微信截图_20240807013716.png
-rw-r--r-- 1 ymq 197121 375342 9月 3 2024 微信截图_20240903103300.png
-rw-r--r-- 1 ymq 197121 312885 9月 6 2024 微信截图_20240906121831.png
drwxr-xr-x 1 ymq 197121 0 3月 12 16:45 应用/

ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ ls -l
total 4644
-rw-r--r-- 1 ymq 197121 162 3月 13 14:51 '~$报告上传问题.docx'
-rwxr-xr-x 1 ymq 197121 976 3月 11 21:08 1yan.lnk*
-rwxr-xr-x 1 ymq 197121 947 3月 11 21:08 1研.lnk*
-rw-r--r-- 1 ymq 197121 282 5月 6 2024 desktop.ini
-rwxr-xr-x 1 ymq 197121 2379 3月 12 08:54 'Microsoft Edge.lnk'*
-rw-r--r-- 1 ymq 197121 901 5月 13 2024 '$'\360\237\232\227'问题汇总.txt'
-rwxr-xr-x 1 ymq 197121 498 3月 13 09:41 百度网盘同步空间.lnk*
-rw-r--r-- 1 ymq 197121 374093 3月 12 21:44 报告上传问题.docx
-rw-r--r-- 1 ymq 197121 559 7月 6 2024 崩铁+原.txt
-rw-r--r-- 1 ymq 197121 69 11月 14 18:36 瓦配置.txt
-rw-r--r-- 1 ymq 197121 1128983 8月 7 2024 微信截图_20240807013607.png
-rw-r--r-- 1 ymq 197121 1307626 8月 7 2024 微信截图_20240807013629.png
-rw-r--r-- 1 ymq 197121 1207682 8月 7 2024 微信截图_20240807013716.png
-rw-r--r-- 1 ymq 197121 375342 9月 3 2024 微信截图_20240903103300.png
-rw-r--r-- 1 ymq 197121 312885 9月 6 2024 微信截图_20240906121831.png
drwxr-xr-x 1 ymq 197121 0 3月 12 16:45 应用/

```

ls -l 与 ls -al 区别,ls -al 等价于 ls -a -l, 后面的短选项可以连写

```

lrwxrwxrwx 1 ymq 197121 53 5月 6 2024 SendTo -> /c/Users/lxm/AppData/Roaming/Micro
soft/Windows/SendTo/
lrwxrwxrwx 1 ymq 197121 56 5月 6 2024 Templates -> /c/Users/lxm/AppData/Roaming/Mi
crosoft/Windows/Templates/
drwxr-xr-x 1 ymq 197121 0 12月 20 19:13 Videos/
drwxr-xr-x 1 ymq 197121 0 3月 13 09:36 'WPS Cloud Files'/
drwxr-xr-x 1 ymq 197121 0 9月 20 16:27 WPSDrive/
drwxr-xr-x 1 ymq 197121 0 3月 7 20:59 Zotero/

ymq@LAPTOP-J3NTE080 MINGW64 ~
$ pwd Templates
/c/Users/lxm

ymq@LAPTOP-J3NTE080 MINGW64 ~
$ cd SendTo

ymq@LAPTOP-J3NTE080 MINGW64 ~/SendTo
$ pwd
/c/Users/lxm/SendTo

```

没有进行跳转

```

ymq@LAPTOP-J3NTE080 MINGW64 ~/Desktop
$ ls -al
total 4664
drwxr-xr-x 1 ymq 197121 0 3月 13 14:51 ./
drwxr-xr-x 1 ymq 197121 0 3月 13 16:51 ../

```



最前'drwxr..'部分为**权限**, drwxr-xr-x 中 r 代表 read, w 代表 write, x 代表 execute 执行, “-rw-r--r—”代表可读可写但不可执行

ls -al 返回内容解析:

各部分含义解析

每行输出分为 7 个字段，从左到右依次为:

1. 文件类型与权限 (drwxr-xr-x)

- 第 1 个字符: 文件类型

符号	含义
-	普通文件 (如文本文件)
d	目录
l	符号链接 (快捷方式)
b	块设备文件
c	字符设备文件

- 后续 9 个字符: 权限模式 (分 3 组)



- r: 可读 (Read)
- w: 可写 (Write)
- x: 可执行 (Execute)
- : 无对应权限

## 2. 硬链接数 ( 1 )

表示该文件/目录的硬链接数量。目录的硬链接数通常至少为 2 (自身 `.` 和父目录 `..`) 。

## 3. 所有者 ( user )

文件/目录的所有者用户名。

## 4. 所属组 ( group )

文件/目录所属的用户组名。

## 5. 大小 ( 4096 )

- 文件：以字节为单位的文件大小。
- 目录：目录元数据占用的磁盘空间 (通常为 4096 字节) 。

## 6. 最后修改时间 ( Jun 10 15:23 )

文件/目录的最后修改日期和时间。

## 7. 名称 ( `.bashrc` 或 `docs -> /mnt/data/docs` )

- 文件/目录的名称。
- 符号链接：会显示 链接名 -> 目标路径 (如 `docs -> /mnt/data/docs`) 。

---

### 特殊条目说明

- `.`：当前目录。
- `..`：上一级目录。
- 以 `.` 开头的文件：隐藏文件 (如 `.bashrc`) 。

`ls -alh,h` 可以体高输出内容可读性,在显示文件大小会用 k,m 等单位

```
-rw-r--r-- 1 ymq 197121 30M 3月 13 00:40 NTUSER.DAT
-rw-r--r-- 1 ymq 197121 2.2M 5月 6 2024 ntuser.dat
-rw-r--r-- 1 ymq 197121 7.3M 5月 6 2024 ntuser.dat
-rw-r--r-- 1 ymq 197121 64K 2月 14 00:12 NTUSER.DAT
e1fe3 TM hlf
```

`ls -alhS,S` 可以使输出时按文件大小排序

其他命令可以从 `ls` 手册中查找选取,或者在 `git bash` 终端中输入 `ls -help` 查看手册

## 任务二,三:

### 任务三

使用 mkdir 在 repo 中批量创建了文件夹 1,2,使用 mkdir -p 在 repo 中创建了 3/3.1/3.1.1 这一文件夹目录:

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mkdir 1 2

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mkdir -P 3/3.1/3.1.1
mkdir: unknown option -- P
Try 'mkdir --help' for more information.

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mkdir -p 3/3.1/3.1.1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls ~/repo/3
3.1/
```

在 repo 中创建了 11.txt 并在其中写入了"text"内容:

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ echo "test" >> 11.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
1/      courses/  test2/    week02/    week2.py
11.txt  test1/    week01/   week1_test1.py  week3.py

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cat 11.txt
test
```

注意语法">>"

在 repo 中创建 22.txt,33.txt

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ touch {22,33}.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
1/      2/      3/      courses/  test2/    week02/    week2.py
11.txt  22.txt  33.txt  test1/    week01/   week1_test1.py  week3.py
```

使用 cp 命令将 11.txt,22.txt 和文件夹 2,3 复制到文件夹 1 中

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cp 2 3 11.txt 22.txt 33.txt 1
cp: -r not specified; omitting directory '2'
cp: -r not specified; omitting directory '3'

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cp -r2 -r3 11.txt 22.txt 33.txt 1
cp: unknown option -- 2
Try 'cp --help' for more information.

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cp -r 2 -r 3 11.txt 22.txt 33.txt 1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
1/      2/      3/      courses/ test2/   week02/   week2.py
11.txt  22.txt  33.txt  test1/   week01/   week1_test1.py week3.py

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls 1
11.txt  2/  22.txt  3/  33.txt
```

第一行代码存在语法错误

用 cp 命令将桌面上的“瓦配置.txt”复制到文件夹 1 和 repo 中

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cp ../Desktop/瓦配置.txt .

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
1/      2/      3/      courses/ test2/   week02/   week2.py 瓦配置.txt
11.txt  22.txt  33.txt  test1/   week01/   week1_test1.py week3.py

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cp ../Desktop/瓦配置.txt 1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls 1
11.txt  2/  22.txt  3/  33.txt 瓦配置.txt
```

用 ls 命令检查文件夹 1 中所有文件/文件夹的大小和修改时间

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls -alht 1
total 6.0K
drwxr-xr-x 1 ymq 197121 0 3月 13 19:47 ./
-rw-r--r-- 1 ymq 197121 69 3月 13 19:47 瓦配置.txt
drwxr-xr-x 1 ymq 197121 0 3月 13 19:45 ../
-rw-r--r-- 1 ymq 197121 0 3月 13 19:43 33.txt
-rw-r--r-- 1 ymq 197121 0 3月 13 19:43 22.txt
-rw-r--r-- 1 ymq 197121 5 3月 13 19:43 11.txt
drwxr-xr-x 1 ymq 197121 0 3月 13 19:43 3/
drwxr-xr-x 1 ymq 197121 0 3月 13 19:43 2/
```

在文件夹 1 中创建文件夹 4

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mkdir -p 1/4
```

将 1 中的几个文件和文件夹移动到文件夹 4

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mv 11.txt 2 22.txt 3 33.txt 1/4

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
1/          test1/  week01/  week1_test1.py  week3.py
courses/   test2/  week02/  week2.py        瓦配置.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls 1
11.txt  2/  22.txt  3/  33.txt  4/  瓦配置.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mv 1/{11.txt 2 22.txt 3 33.txt} 1/4
mv: cannot stat '1/{11.txt': No such file or directory
mv: cannot stat '2': No such file or directory
mv: cannot stat '22.txt': No such file or directory
mv: cannot stat '3': No such file or directory
mv: cannot stat '33.txt}': No such file or directory

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mv /1/{11.txt 2 22.txt 3 33.txt} /1/4
mv: target '/1/4' is not a directory

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mv /1/{11.txt 2 22.txt 3 33.txt} 1/4
mv: cannot stat '/1/{11.txt': No such file or directory
mv: cannot stat '2': No such file or directory
mv: cannot stat '22.txt': No such file or directory
mv: cannot stat '3': No such file or directory
mv: cannot stat '33.txt}': No such file or directory

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mv 1/{11.txt,2,22.txt,3,33.txt} 1/4

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls 1
4/ 瓦配置.txt
```

用 rm 删除文件夹 1

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ rm 1
rm: cannot remove '1': Is a directory

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ rm -r 1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ ls
courses/  test1/  test2/  week01/  week02/  week1_test1.py  week2.py  week3.py
```

第一行的语法是在 1 为文件时使用,想要删除文件夹也就是目录需要添加-r

# 命令 cp

在 Git Bash 中使用 `cp` 命令将其他目录下的文件复制到当前工作目录时，可以按照以下格式输入命令：

```
bash
cp /其他目录的路径/文件名 .
```

## 关键点解释：

1. `.` **表示当前目录**：命令末尾的 `.` 代表当前工作目录。
2. **路径格式**：如果路径中包含空格或特殊字符，需要用引号包裹路径（或使用转义符 `\`）。

## 示例：

假设你想将 `C:\Users\YourName\Documents\example.txt` 复制到当前目录，命令如下：

```
bash
cp "/c/Users/YourName/Documents/example.txt" .
```

命令: `cp [文件路径(来源)] .(当前工作目录)`

# 命令:mv

## 一、基本用法

### 1. 移动文件

bash

复制

```
mv 源文件路径 目标路径
```

示例:

bash

复制

```
# 将当前目录下的 file.txt 移动到上级目录
mv file.txt ..

# 将 /c/Users/文档/example.txt 移动到当前目录
mv /c/Users/文档/example.txt .
```

### 2. 重命名文件

bash

复制

```
mv 原文件名 新文件名
```

示例:

bash

复制

```
# 将 old.txt 重命名为 new.txt
mv old.txt new.txt
```

### 3. 移动并重命名

bash

复制

```
mv 源文件路径 目标路径/新文件名
```

示例:

bash

复制

```
# 将当前目录的 image.jpg 移动到 /images 目录并重命名为 photo.jpg
mv image.jpg images/photo.jpg
```



## 二、常用参数

### 1. 强制覆盖（忽略警告）

使用 `-f` (force) 参数直接覆盖目标文件，不提示确认：

```
bash 复制
mv -f 源文件 目标路径
```

### 2. 交互模式（确认覆盖）

使用 `-i` (interactive) 参数，覆盖前要求确认：

```
bash 复制
mv -i 源文件 目标路径
```

### 3. 不覆盖已有文件

使用 `-n` (no-clobber) 参数，若目标文件已存在，则取消操作：

```
bash 复制
mv -n 源文件 目标路径
```

## 三、常见场景处理

### 1. 移动多个文件

```
bash 复制
mv file1.txt file2.jpg /目标目录/
```

### 2. 批量移动文件（通配符）

```
bash 复制

# 移动所有 .txt 文件到 docs 目录
mv *.txt docs/

# 移动以 "temp" 开头的文件到 backup 目录
mv temp* backup/
```

### 3. 移动目录

直接使用目录名（无需额外参数）：

bash复制

```
mv 原目录名 新目录名      # 重命名目录
mv 原目录名 目标路径/      # 移动目录
```

### 4. 处理文件名中的空格

用引号包裹文件名或使用转义符 `\`：

bash复制

```
mv "my file.txt" ../documents/
mv my\ file.txt ../documents/
```

## 四、注意事项

- 1. **不可逆操作**：`mv` 会直接覆盖目标路径的同名文件（除非使用 `-i` 或 `-n`）。
- 2. **权限问题**：确保对源文件有读取权限，对目标路径有写入权限。
- 3. **路径格式**：
  - Windows 绝对路径需转换为 Git Bash 格式（如 `C:\Users` → `/c/Users`）。
  - 相对路径用 `.`（当前目录）或 `..`（上级目录）表示。
- 4. **区分大小写**：Git Bash 的路径和文件名是大小写敏感的。

## 五、总结

- **移动文件/目录** → `mv 源路径 目标路径`
- **重命名文件/目录** → `mv 旧名 新名`
- **覆盖控制**：`-f`（强制）、`-i`（确认）、`-n`（不覆盖）
- **批量操作**：结合通配符 `*` 或 `{}`（例如 `mv *. {txt,jpg} folder/`）

在 Git Bash 中，将多个非当前工作目录下的文件移动到第三个目录，可以通过以下步骤实现：

### 关键语法

bash复制

```
mv 源路径/文件1 源路径/文件2 ... 目标目录路径/
```

## 具体步骤

### 1. 明确路径关系

假设：

- 源文件路径： `/path/to/source/files/`
- 目标目录路径： `/path/to/destination/`

### 2. 命令格式

```
bash 复制
mv /path/to/source/files/{file1,file2,file3} /path/to/destination/
```

或直接列出所有文件名：

```
bash 复制
mv /path/to/source/file1.txt /path/to/source/file2.jpg /path/to/destination/
```

## 场景示例

### 示例 1：移动特定文件

将 `~/Documents/reports` 目录下的 `report1.pdf` 和 `report2.pdf` 移动到 `/c/Users/YourName/Desktop/backup`：

```
bash 复制
mv ~/Documents/reports/report{1,2}.pdf /c/Users/YourName/Desktop/backup/
```

### 示例 2：使用通配符批量移动

将 `/c/Projects/images` 目录下的所有 `.jpg` 文件移动到 `/c/Users/YourName/Pictures`：

```
bash 复制
mv /c/Projects/images/*.jpg /c/Users/YourName/Pictures/
```

### 示例 3：移动多个不同扩展名的文件

将 `/c/Downloads` 下的 `notes.txt` 和 `diagram.png` 移动到 `/d/Archive`：

```
bash 复制
mv /c/Downloads/notes.txt /c/Downloads/diagram.png /d/Archive/
```

# 命令 mkdir

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ mkdir 1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cd 1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/1
$ ls

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/1
$ ls -a
./ ../

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/1
$
```

## 基本语法

bash

复制

`mkdir` 文件夹名

## 示例

bash

复制

```
# 创建一个名为 "docs" 的文件夹
mkdir docs

# 创建多个文件夹（多个名称用空格分隔）
mkdir images css js
```

## 常见场景处理

### 1. 创建嵌套的多级目录

使用 `-p` 参数，可以一次性创建多层目录（如果父目录不存在，会自动创建）：

```
bash
# 创建多级目录（例如 src/main/java）
mkdir -p src/main/java
```

复制

### 2. 文件夹名包含空格

用引号包裹名称或使用转义符 `\`：

```
bash
mkdir "my folder"
mkdir my\ folder
```

复制

### 3. 在指定路径下创建文件夹

```
bash
# 在上级目录创建文件夹
mkdir ../backup

# 在绝对路径下创建（例如 C:\Users\YourName\Desktop）
mkdir /c/Users/YourName/Desktop/temp
```

复制

## 总结

- 基础创建 → `mkdir 文件夹名`
- 创建多级目录 → `mkdir -p 路径/子路径`

# 命令 rm

## 基本语法

bash

复制

```
rm 文件名
```

## 示例

bash

复制

```
# 删除当前目录下的 example.txt
rm example.txt

# 删除多个文件（用空格分隔文件名）
rm file1.txt file2.jpg
```

## 常用场景和参数

### 1. 强制删除（忽略警告）

如果文件有只读属性或需要跳过确认提示，使用 `-f` (force) 参数：

bash

复制

```
rm -f 文件名
```

### 2. 删除文件名包含空格或特殊字符的文件

用引号包裹文件名或使用转义符 `\`：

bash

复制

```
rm "my file.txt"
rm my\ file.txt
```

### 3. 批量删除文件（通配符）

使用 `*` 通配符删除符合模式的文件：

bash

复制

```
# 删除所有 .tmp 文件
rm *.tmp

# 删除所有以 "temp" 开头的文件
rm temp*
```

### 4. 删除隐藏文件（以 `.` 开头的文件）

bash

复制

```
rm .hiddenfile
```

### 扩展：删除文件夹

如果需要删除目录（文件夹），需使用 `-r` 参数（递归删除）：

```
bash 复制  
rm -r 文件夹名
```

### 总结

- 删除单个文件 → `rm 文件名`
- 强制删除 → `rm -f 文件名`
- 批量删除 → `rm *.扩展名`
- 慎用通配符：`rm *` 会删除当前目录下所有文件！

# 命令 df

## 2. 常用选项

以下是一些常用选项（部分选项可能与 Linux 版本不同）：

显示人类可读的格式（如 KB、MB、GB）

```
bash
```

复制

```
df -h
```

显示文件系统类型

```
bash
```

复制

```
df -T
```

仅显示特定文件系统类型

例如，显示 NTFS 文件系统：

```
bash
```

复制

```
df -T | grep ntfs
```

汇总所有文件系统的总使用情况

```
bash
```

复制

```
df --total
```





## 排除特定文件系统类型

例如，排除临时文件系统：

```
bash
df -x tmpfs
```

复制

## 3. 过滤输出

由于 Git Bash 基于 Windows，文件系统的路径可能与 Linux 不同（如 `/c` 对应 `C:\`）。可以通过以下方式过滤结果：

查看特定磁盘（如 C 盘）

```
bash
df -h /c
```

复制

使用 `grep` 过滤关键词

```
bash
df -h | grep "Filesystem\\|/c"
```

复制

## 5. 示例

查看所有磁盘的使用情况（人类可读格式）

```
bash
df -h
```

复制

仅显示 C 盘和 D 盘的信息

```
bash
df -h /c /d
```

复制

显示文件系统类型并过滤 NTFS

```
bash
df -T | grep ntfs
```

复制

# 命令 du

## 3. 常用命令选项

选项	说明	示例
-h	以人类可读格式显示 (如 KB, MB, GB)	du -h filename
-s	仅显示总大小 (不递归子目录)	du -sh /path/to/dir
--max-depth=N	限制显示的目录深度	du -h --max-depth=1
-a	显示所有文件 (包括子目录中的文件)	du -ah

## 4. 实用示例

bash复制

```
# 1. 查看当前目录总大小
du -sh .

# 2. 查看当前目录下各子目录的大小 (深度1)
du -h --max-depth=1

# 3. 查看特定文件/目录的大小
du -sh /c/Users/YourName/Documents

# 4. 按大小排序 (配合 sort)
du -h --max-depth=1 | sort -hr
```

▼

其他格式命令示例：  
du -s \* | sort -nr > ~/report.txt (详细见任务五)

## 任务四:用 df 查看磁盘剩余空间，使用 du 命令查看文件/文件夹占用的磁盘空间

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
D:/Lxm/Git      288059388 236169220   51890168   82% /
C:              209716220 152166412   57549808   73% /c
```

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ df /d
Filesystem      1K-blocks      Used Available Use% Mounted on
D:              288059388 236257216   51802172   83% /d
```

两次命令的结果显示 d 盘出现了两种名称（大小相同），原因如下：

### 1. 根本原因：挂载点（Mount Point）的差异

在 Git Bash 中，Windows 磁盘默认会挂载到类似 `/c`、`/d` 的路径，但某些情况下可能存在 **自定义挂载** 或 **嵌套挂载**。你的输出表明：

- 第一次 `df` 命令：

`D:/Lxm/Git` 被挂载到了根目录 `/`，因此显示为：

```
D:/Lxm/Git -> Mounted on /
```

这可能是因为你的 Git 仓库或某个脚本将 `D:/Lxm/Git` 目录直接挂载到了根目录（例如通过 `mount` 命令或环境变量配置）。

- 第二次 `df /d` 命令：

直接查询 `/d` 路径时，显示的是默认的 D 盘挂载点：

```
D: -> Mounted on /d
```

两个挂载点（`/` 和 `/d`）实际指向同一个物理磁盘（D 盘），因此磁盘使用情况（1K-blocks、Used 等）完全一致。

- **Git 仓库的特殊配置**：如果你在 `D:/Lxm/Git` 目录下初始化了 Git 仓库，某些工具（如 Git for Windows）可能会自动调整挂载点。
- **安装时的路径选择**：在安装 Git for Windows 时，如果选择了非默认路径（如 `D:/Lxm/Git`），可能导致根目录挂载到该路径。

## 4. 对实际使用的影响

- 路径一致性:

虽然根目录 `/` 和 `/d` 都指向 D 盘, 但它们的“逻辑路径”不同:

- 在 `/` 下操作文件, 实际对应 `D:/Lxm/Git`。
- 在 `/d` 下操作文件, 实际对应 `D:` 的根目录。

- 潜在问题:

如果根目录 `/` 被挂载到 `D:/Lxm/Git`, 可能导致某些依赖绝对路径的脚本或命令行为异常 (例如访问 `/home` 时实际指向 `D:/Lxm/Git/home`)。

---

## 5. 如何恢复默认挂载?

如果需要将根目录 `/` 恢复为默认挂载 (通常挂载到 MSYS2 的安装目录), 可以:

1. 打开 Git Bash 的配置文件 (如 `C:\Program Files\Git\etc\profile` 或用户目录的 `.bashrc`)。
2. 删除或注释掉自定义的 `mount` 命令。
3. 重启 Git Bash。

但当前配置不影响工作 (例如 Git 仓库路径正常且无权限问题), 可以忽略挂载点差异。

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ du -h --max-depth=2 ..
0      ../anaconda
2.0K   ../conda
1.0K   ../continuum/anaconda-client
1.0K   ../continuum
45K    ../douyu_channel/logs
45K    ../douyu_channel
0      ../idlerc
57K    ../ipython/profile_default
57K    ../ipython
7.0K   ../ssh
0      ../vscode/cli
169M   ../vscode/extensions
169M   ../vscode
1.0K   ../xp2p
972K   ../ado/plus
972K   ../ado
15M    ../AppData/ACLOS
du: cannot read directory '../AppData/Local/ElevatedDiagnostics': Permission denied
du: cannot read directory '../AppData/Local/Steam/htmlcache/WidevineCdm/4.10.2830.0': Permission denied
du: cannot read directory '../AppData/Local/Temp/tmpigxxt26p': Permission denied
9.6G   ../AppData/Local
4.0G   ../AppData/LocalLow
6.9G   ../AppData/Roaming
21G    ../AppData
1.0K   ../Contacts
135K   ../Desktop/应用
4.7M   ../Desktop
0      ../Documents/Adobe
```

## 任务五:

解释命令: `du -s * | sort -nr > ~/report.txt`

这是一个命令行指令。“`du -s *`”用于显示当前目录下每个文件和目录的磁盘使用情况汇总。“`|`”是管道符号,将前面命令的输出作为后面命令的输入。“`sort -nr`”表示对输入内容按照数字大小进行降序排序。“`>`”是重定向符号,将排序后的结果输出到“`~/report.txt`”文件中,即将当前目录下文件和目录的磁盘使用情况汇总后进行降序排序,并保存到用户主目录下的“`report.txt`”文件中。

```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ du -s * | sort -nr > ~/report.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cat ../report.txt
13577    courses
1258     week01
665      week02
4        week1_test1.py
0        week3.py
0        week2.py
0        test2
0        test1

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ du -h --max-depth=1 | sort -n > ~/report2.txt

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo
$ cat ../report2.txt
0        ./test1
0        ./test2
1.3M     ./week01
14M      ./courses
16M      .
665K     ./week02
```

\*为通配符,用于选取当前目录下所有文件

`du -h --max-depth=1 | sort -n > ~/report2.txt` 这一命令只查看了当前目录下深度为1的子目录的大小,并没有查看当前目录下的非文件夹的文件大小,且在添加-h后其按数字排序的方式并没有考虑单位。

## 任务六:

在 GitCode 平台新建一个你个人的私密的代码仓库 (非公开, 别人看不到), clone 到本地, 将一些你自己的工作文件 (文本文件或二进制文件都可以) 添加到仓库里, push 到平台上托管



```
ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/ymq-work (main)
$ git add 学习日记.pdf

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/ymq-work (main)
$ git commit -m"第一周笔记"
[main ff4d4eb] 第一周笔记
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "\345\255\246\344\271\240\347\254\224\350\256\260.pdf"

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/ymq-work (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 541.54 KiB | 17.47 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Start Git Hooks Checking [PASSED]
To gitcode.com:y-m-q/ymq-work.git
74f0bbf..ff4d4eb main -> main

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/ymq-work (main)
$ git log
commit ff4d4eb0a6bb3584500a47fc66ba8aa19f639cac (HEAD -> main, origin/main, origin/HEAD)
Author: y-m-q <18210710168@163.com>
Date: Thu Mar 13 21:51:10 2025 +0800

    第一周笔记

commit 74f0bbf88a9308b58c322d95978cd86f29d2c1fe
Author: y-m-q <18210710168@163.com>
Date: Thu Mar 13 21:42:19 2025 +0800

    Initial commit

ymq@LAPTOP-J3NTE080 MINGW64 ~/repo/ymq-work (main)
$
```

Y

y-m-q/  
ymq-work

🔍

7

Y

🔍 2 提交数

🔄 0

★ 0

🔗 0 ▾

📄

📁

🔴 代码

🔵 Issues

🔗 Pull Requests

💬 讨论

👤 项目成员 1

📊 分析

⚙️ 项目设置

main ▾

+

🔍 搜索文件

Y y-m-q 第一周笔记

ff4d4eb0 | 创建于 1 分钟前 | 🕒

文件	最后提交记录	最后更新时间
📄 README.md	Initial commit	9 分钟前
📄 学习笔记.pdf	第一周笔记	1 分钟前