金融计算机语言第二周作业

专题一 理解 Unix 路径

Unix 路径是 Unix 操作系统中用于标识文件或目录在文件系统中位置的字符串。

1. 目录树

目录树是一种以树状结构来展示文件系统中目录和文件层次关系的可视化表示方法。管理文件系统资源过程中采取树形及结构，有很多节点。

2. 根目录

根目录是文件系统目录结构的起始点和基础，就像一棵树的根部，所有其他目录和文件都从根目录分支展开。

在 Unix、Linux 等类 Unix 系统中，根目录用正斜杠 “/” 表示。在 Windows 系统中，根目录通常用盘符加反斜杠表示，如 “C:\”“D:\” 等，其中 “C:”“D:” 等是不同的磁盘分区，反斜杠 “\” 表示该分区的根目录。

```
86150@GFJ MINGW64 ~/Documents
$ pwd
/c/Users/86150/Documents
```

```
86150@GFJ MINGW64 ~/Documents
$ cd /

86150@GFJ MINGW64 /
$ ls
bin/  dev/  git-bash.exe*  LICENSE.txt  proc/          tmp/       unins000.exe*  usr/
cmd/  etc/  git-cmd.exe*   mingw64/     ReleaseNotes.html  unins000.dat  unins000.msg
```

3. 路径

路径是指用于标识文件或目录在文件系统中位置的字符串。在命令行中要访问某一文件或其他程序时，需要给出相应路径。

应用程序、代码、数据都在文件系统里，这些都属于计算机的资源。计算机在使用它们时，需要通过路径进行区别。

- **相对路径**

相对路径是相对于当前工作

目录的路径。它不包含从根目录开始的完整路径信息，而是从当前所在目录出发来描述文件或目录的位置。

- **绝对路径**

绝对路径是从根目录开始的完整路径，它明确地指定了文件或目录在文件系统中的精确位置。

```
86150@GFJ MINGW64 /
$ pwd
/

86150@GFJ MINGW64 /
$ cat abc.txt                    相对路径
cat: abc.txt: No such file or directory

86150@GFJ MINGW64 /                      绝对路径
$ cat /c/Users/86150/Desktop/abc.txt
hello
```

4. 分隔符 “/”

```
86150@GFJ MINGW64 ~/Documents
$ pwd
/c/Users/86150/Documents
```

5. Unix 路径的标准写法

-Unix 路径

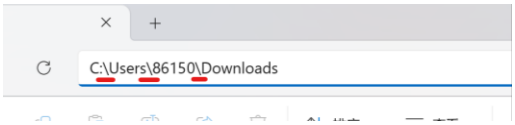
① 绝对路径以根目录（/）开头

② 以斜杠（/）作为路径分隔符

```
86150@GFJ MINGW64 ~/Downloads
$ pwd
/c/Users/86150/Downloads
```

-Windows 路径

反斜杠（\）且无根目录，从盘符开始“c:”



p. s. 浏览器中使用斜杠而非反斜杠是因为地址网址是在互联网上，而互联网是一个开放的标准，其标准遵循 Unix 的标准。

6. ls 命令

显示当前工作目录下的文件。

ls 是 “list” 的缩写，主要用于列出目录内容，包括文件和子目录。它可以帮助用户查看当前

目录下有哪些文件和文件夹，也可以查看指定目录下的内容。

使用 ls 命令，检查到本机的“桌面”、“下载”、“文档”等常用文件夹的真实文件系统路径是 c 盘 Users/86150 中。

```
Desktop/
Documents/
Downloads/
Favorites/
Links/
```

7. pwd 命令

```
86150@GFJ MINGW64 ~
$ pwd
/c/Users/86150
```

显示当前工作目录。

pwd 是“print working directory”的缩写，用于显示当前工作目录的绝对路径。当你在终端中进行操作时，知道当前所在的目录位置非常重要，pwd 命令可以确认这一点。

8. cd 命令

使用 **tab 键**可以在命令中自动补全。

每一命令行后的“~”为用户主目录。

```
86150@GFJ MINGW64 ~
$ cd Desktop/
```

cd 命令是 Unix、Linux 等类 Unix 系统中一个极为基础且重要的命令，全称为“change directory”，其主要功能是用于切换当前工作目录。

```
86150@GFJ MINGW64 ~
$ cd Desktop/

86150@GFJ MINGW64 ~/Desktop
$ pwd
/c/Users/86150/Desktop

86150@GFJ MINGW64 ~/Desktop
$ ls
 abc.txt                'EViews10_x64 - 快捷方式.lnk'*
'CAJViewer 8.0.lnk'*     'Google Chrome.lnk'*
'Counter-Strike 2.url'   "It Takes Two Friend's Pass.url"
 desktop.ini              M-AI.lnk*

86150@GFJ MINGW64 ~/Desktop
$ cat abc.txt
hello
```

cd ..命令[1]：返回上级文件夹。

[1] 命令后加“.”表示当前目录；命令后加“..”表示上级目录。

```
86150@GFJ MINGW64 ~/Desktop
$ cd ..

86150@GFJ MINGW64 ~
$ ls
 「开始」菜单@          NetHood@
 AppData/              NTUSER.DAT
'Application Data'@    ntuser.dat.LOG1
 Contacts/             ntuser.dat.LOG2
 Cookies@              NTUSER.DAT{41d29057-4f7f-11ef
 Desktop/              NTUSER.DAT{41d29057-4f7f-11ef
 Documents/            NTUSER.DAT{41d29057-4f7f-11ef
 Downloads/            ntuser.ini
 Favorites/            OneDrive/
 Links/                Pictures/
'Local Settings'@      PrintHood@
 Music/                PycharmProjects/
'My Documents'@        Recent@
```

cd ../文件名/：返回到上级文件夹下的另一文件。

```
86150@GFJ MINGW64 ~
$ cd Downloads/

86150@GFJ MINGW64 ~/Downloads
$ pwd
/c/Users/86150/Downloads

86150@GFJ MINGW64 ~/Downloads
$ cd ../Documents/

86150@GFJ MINGW64 ~/Documents
$ ls
 Adobe/                 'EViews Addins'/
'AI Meeting Manager'/   'EViews User Objects'/
'AI Recorder'/           KingsoftData/
 desktop.ini             MiMouseAI/
```

专题二 理解 shell 命令行

Shell 基本语法结构

Shell 是一种命令行解释器，它提供了用户与操作系统内核之间的接口。命令行也是一种程序，有一定语法规则，将命令行中的指令翻译给操作系统。包括 Bash、Zsh 等。(Bash&Zsh 兼容)

1. 空格分隔

空格即命令行语法解析时的分隔符号。

```
86150@GFJ MINGW64 ~
$ ls repo
script1.py  week01/
```

2. 短选项

短选项通常由一个连字符“-”后面跟一个单个字符组成。

3. 长选项

长选项由两个连字符“--”后面跟一个或多个单词组成，单词之间可以使用连字符“-”连接。

```
-a, --all 长
短     do not ignore entries starting with .
```

4. 参数

参数指的是传递给函数、程序或者命令的额外信息，目的是对其行为进行控制和定制。

```
86150@GFJ MINGW64 ~
$ ls repo
script1.py  week01/
```

5. ls 命令的选项

https://man7.org/linux/man-pages/man1/ls.1.html

SYNOPSIS top

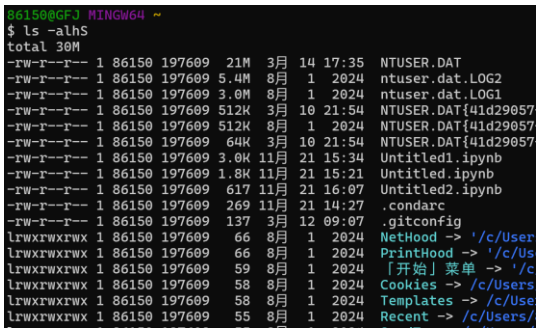
ls [*OPTION*]... [*FILE*]...

ls 常用选项：

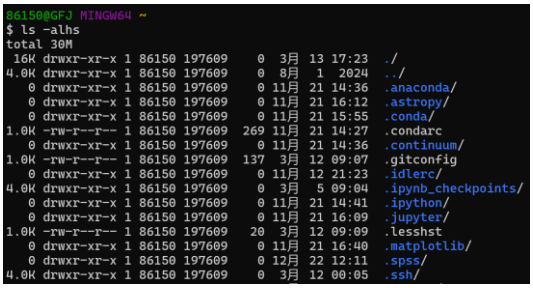
-a：显示所有选项

-l：use a long listing format

-S：排序（按文件大小）



-t：排序（按修改时间）

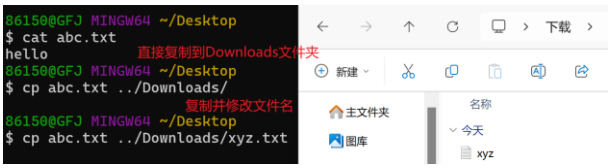


专题三 常用命令与大模型解释

一、 常用命令

1. cp 命令（copy）

cp 命令用于复制文件或目录。

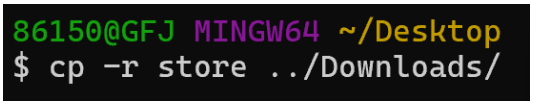


命令中第一个路径为文件来源，第二个路径为复制地址。

复制文件夹时：

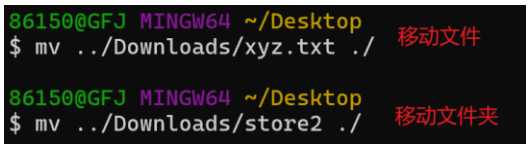
-r：recursively 递归

正确命令行为:



2. mv 命令（move）

mv 命令既可以移动文件或目录，也能对文件或目录进行重命名操作。



在 mv 命令中不需要使用-r 的选项，是因为此命令未使用递归算法，只是修改了文件的路径名称。

mv 命令下的移动，不涉及磁盘上数据的移动，只是将磁盘上的文件修改了名称。移动的是磁盘中文件的节点 inode，把文件节点的信息进行修改，修改成新路径。

3. mkdir 命令（make directory）

mkdir 是用于创建目录（文件夹）的基础命令。

4. rm 命令（remove）

rm 命令是用于删除文件或目录的常用命令

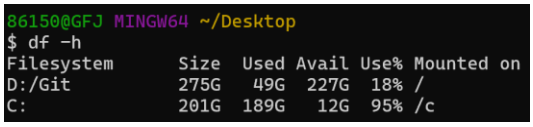
rm 命令使用递归算法，需要使用-r 选项。当删除某些无写入权限的文件/文件夹时，需要使用-rf 来强制删除。

如：删除根目录

e.g. 用 mkdir 命令创建一个 myproject 文件夹，复制一些文件进文件夹，用 ls 命令查看这些文件/文件夹的大小或修改时间，最后用 rm 命令删除它们。

5. df 命令（disk free）

df 命令是用于查看磁盘空间使用情况的基础命令。



6. du 命令（disk usage）

du 命令是用于估算文件和目录磁盘使用量的命令。

二、 用大模型解释命令

du -s * | sort -nr > ~/report.txt

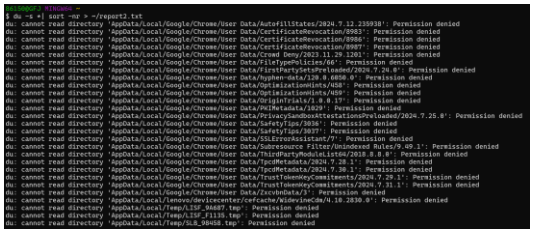
-总体解释：计算当前目录下所有文件和目录的磁盘使用量，按照磁盘使用量的数值大小进行降序排序，然后将排序后的结果保存到用户主目录下的 report.txt 文件中。

-分解：

- -s 选项即--summarize，它的作用是仅显示每个参数的总计信息，也就是只输出指定文件或目录的总磁盘使用量，而不会显示其下子目录和文件的详细磁盘使用情况。
- *是一个通配符，代表当前目录下的所有文件和目录。
- |是管道符号，它的作用是将前一个命令的（du -s *）的标准输出作为最后一个命令（sort -nr）的标准输入。这样，du -s *统计得到的文件和目录大小信息就会被传递给 sort -nr 命令进行处理。
- 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，而不是按照字典顺序排序。

- -r 选项表示以逆序（降序）排列。
- >是输出重定向符号，它的作用是将前面命令的输出结果覆盖写入到指定的文件中。
- ~/report.txt 表示用户主目录下的 report.txt 文件。

实践：



在操作过程中出现了 permission denied 的情况，通过大模型搜索，发现该情况的发生是由于 du 命令在尝试访问某些文件或目录时没有足够的权限。

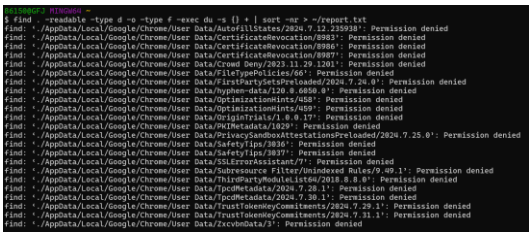
有以下几种解决办法：

1. 使用 sudo 以管理员权限执行命令

```
86150@GFJ MINGW64 ~
$ sudo du -s * | sort -nr > ~/report.txt
bash: sudo: command not found
```

系统中未安装相关命令。

2. 找出无权限的文件或目录并跳过



权限依旧受限。

3. 手动检查并修改权限

```
bash
chmod +r /path/to/directory_or_file
```

4. 限制 du 命令的搜索范围

```
86150@GFJ MINGW64 ~
$ du -s / | sort -nr > ~/report.txt
du: cannot read directory '/tmp/LISF_9A687.tmp': Permission denied
du: cannot read directory '/tmp/LISF_F1135.tmp': Permission denied
du: cannot read directory '/tmp/SLB_98458.tmp': Permission denied
```

当在子目录下，如：desktop 文件夹下，可以生成 report 文件。

```
86150@GFJ MINGW64 ~
$ cd Desktop/

86150@GFJ MINGW64 ~/Desktop
$ du -s * | sort -nr > ~/report.txt
```

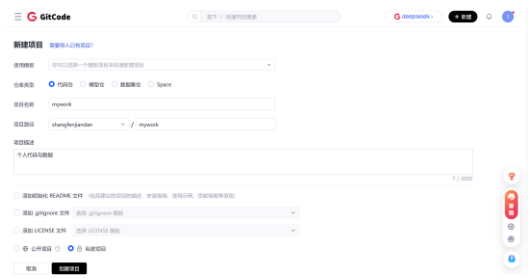
C:\Users\86150\Desktop\report.txt

专题四 用私密仓库托管自己的文件

- 建立个人代码仓库

私密的个人代码仓库可以创建多个，用于不同方面，托管不同数据。可以保存本地数据，防止由于电脑损坏导致的数据丢失。代码仓库主要保存文本数据，也可以保存少量图片。

1.创建代码仓库




```
86150@GFJ MINGW64 ~
$ cd repo

86150@GFJ MINGW64 ~/repo
$ git clone git@gitcode.com:shangfenjiandan/mywork.git
Cloning into 'mywork'...
warning: You appear to have cloned an empty repository.

86150@GFJ MINGW64 ~/repo
$ ls -l
total 5
drwxr-xr-x 1 86150 197609   0  3月 18 09:25 mywork/
-rw-r--r-- 1 86150 197609 261  3月 11 23:42 script1.py
drwxr-xr-x 1 86150 197609   0  3月 12 00:27 week01/

86150@GFJ MINGW64 ~/repo
$ cd mywork

86150@GFJ MINGW64 ~/repo/mywork (main)
$ ls -l
total 0

86150@GFJ MINGW64 ~/repo/mywork (main)
$ git log
fatal: your current branch 'main' does not have any commits yet
```

2.复制文件

```
86150@GFJ MINGW64 ~/repo/mywork (main)
$ cp ~/repo/script1.py ./

86150@GFJ MINGW64 ~/repo/mywork (main)
$ ls -al
total 5
drwxr-xr-x 1 86150 197609   0  3月 18 09:35 ./
drwxr-xr-x 1 86150 197609   0  3月 18 09:25 ../
drwxr-xr-x 1 86150 197609   0  3月 18 09:25 .git/
-rw-r--r-- 1 86150 197609 261  3月 18 09:35 script1.py
```

3.提交文件（本地代码仓库）

```
86150@GFJ MINGW64 ~/repo/mywork (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script1.py

nothing added to commit but untracked files present (use "git add" to track)

86150@GFJ MINGW64 ~/repo/mywork (main)
$ git add .

86150@GFJ MINGW64 ~/repo/mywork (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   script1.py
```

```
86150@GFJ MINGW64 ~/repo/mywork (main)
$ git commit -m "added some files for test"
[main (root-commit) 93b7080] added some files for test
 1 file changed, 7 insertions(+)
 create mode 100644 script1.py

86150@GFJ MINGW64 ~/repo/mywork (main)
$ git log
commit 93b7080208ff0638c372ecd962fdda906d4b268b (HEAD -> main)
Author: shangfenjiandan <shangfenjiandan@noreply.gitcode.com>
Date:   Tue Mar 18 09:37:59 2025 +0800

    added some files for test
```

4.推送平台

```
86150@GFJ MINGW64 ~/repo/mywork (main)
$ git  push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 381 bytes | 127.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Start Git Hooks Checking                                    [PASSED]
To gitcode.com:shangfenjiandan/mywork.git
 * [new branch]      main -> main
```

5.删除文件

```
86150@GFJ MINGW64 ~/repo/mywork (main)
$ rm ./script1.py
```

6.修改文件

7.使用 cp 命令重新操作