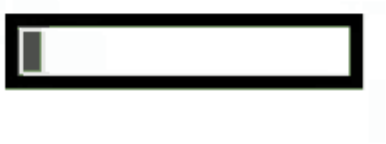
实验课 06

实验6-1 Python中的日期与时间

下载文件时我们最盼望的就是**进度条走得快一点！！！**



下面这段代码就是用Python编写一个模拟进度条的小程序，你可以尝试运行一下并阅读代码。*(代码摘自菜鸟教程)*

```
import time

scale = 50

print("执行开始".center(scale//2,"-"))  # .center() 控制输出的样式，汉字居中，两侧填充 -

start = time.perf_counter() # 调用一次 perf_counter()，从计算机系统里随机选一个时间点A，计算其距离当前时间点B1有多少秒。当第二次调用该函数时，默认从第一次调用的时间点A算起，距离当前时间点B2有多少秒。两个函数取差，即实现从时间点B1到B2的计时功能。
for i in range(scale+1):
    a = '*' * i              # i 个长度的 * 符号
    b = '.' * (scale-i)  # scale-i） 个长度的 . 符号。符号 * 和 . 总长度为50
    c = (i/scale)*100  # 显示当前进度，百分之多少
    dur = time.perf_counter() - start     # 计时，计算进度条走到某一百分比的用时
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,dur),end='')  # \r用来在每次输出完成后，将光标移至行首，这样保证进度条始终在同一行输出，即在一行不断刷新的效果；{:^3.0f}，输出格式为居中，占3位，小数点后0位，浮点型数，对应输出的数为c；{}，对应输出的数为a；{}，对应输出的数为b；{:.2f}，输出有两位小数的浮点数，对应输出的数为dur；end=''，用来保证不换行，不加这句默认换行。
    time.sleep(0.1)      # 在输出下一个百分之几的进度前，停止0.1秒
print("\n"+"执行结束".center(scale//2,'-'))
```

步骤1 获取日期时间

获取当前时间的最简单方式就是使用time库中的`time()`方法。

别忘了首先导入time库。

```
import time
t1 = time.time()
print(t1)
```

运行结果并不是我们熟悉的例如08:32:15这种形式，实际上直接使用`time()`方法返回的是当前时刻与**1970年1月1日0时0分0秒**的差距秒数。

可能你曾听说过“千年虫”问题，不过接下来十几年在计算机方面人们可能需要关心的是2038问题，你可以点击这两个维基百科的链接看一下计算机与时间的爱恨情仇。

刚才我们看到的时间显示方式并不太友好，我们可以使用下面的一些方法来获得以符合人们阅读习惯的方式的时间。

```
t2 = time.localtime()
print(t2)
```

现在看来大概能看出当前日期和时间，不过返回的是一个“散列”的元组，可以供我们单独提取年、月、日等信息使用。

我们还可以以更友好地方式显示时间！

```
t3 = time.asctime(time.localtime())
print(t3)
```

这看起来舒服多了，效果就像是一些网页头部显示的当前时间。

当然我们可以自定义各部分的输出顺序！

```
t4 = time.strftime("%Y/%m/%d %H:%M:%S",time.localtime())
print(t4)
```

> 小结一下我们刚才使用到的time库中的常用方法：
>
> - `time()` 获取UNIX时间
> - `localtime()` 获取当前时刻的结构化元组
> - `asctime()` 获取格式化的时间
> - `strftime()` 返回自定义格式的时间字符串

步骤2 休眠

刚才的模拟进度条小程序中有一行代码`time.sleep(0.1)`

你可以把该行代码注释掉，然后再次运行该小程序，你会发现进度条瞬间抵达终点。

我们知道目前计算机运行速度极快，有的时候需要让计算机主动“等一等”我们。例如很多网站对恶意爬取采取限制措施，在设计爬虫程序时，为了不被远程服务器误认为恶意爬虫，我们可以故意设置两次爬取之间的时间间隔。

time库中提供了`sleep()`方法让当前线程推迟运行。该方法需要的参数为推迟运行的时间，单位为秒，当然你也可以推迟运行小于1秒的时间，不过受机器精度限制建议最小延迟时间为50ms。

```
print('当前时间为：',time.asctime())
time.sleep(3)  # 命令线程延迟运行3秒
print('当前时间为：',time.asctime())
```

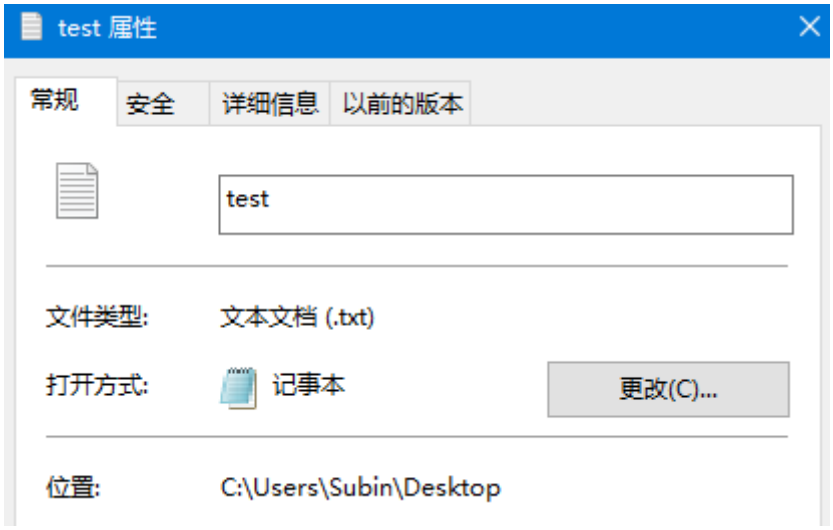
实验6-2 Python文件操作初步

获取数据可以从网络上在线获得，更多的时候需要读取存储在硬盘上的文件。同样，一些程序的运行结果需要持久存储在硬盘中，否则程序运行结束或计算机断电重启后我们需要再次运行程序，这对于大规模程序来说，既耗费时间又占用系统资源。所以我们需要对文件的读写操作有初步的认识。

步骤1 绝对路径和相对路径

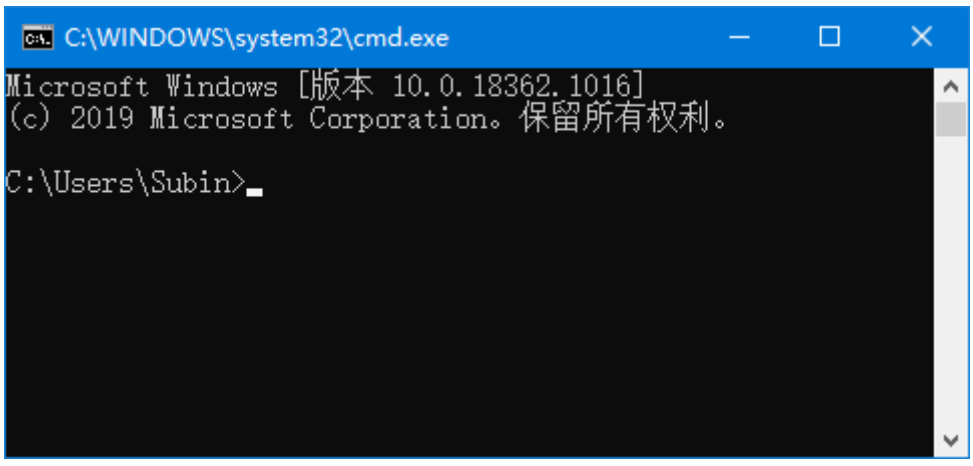
就像我们找一家商店需要知道它的门牌号码一样，寻找一个文件也需要知道它的存储位置。无论在Windows系统还是Linux系统，文件的路径可用绝对路径和相对路径两种方式表示。

例如我们在Windows系统的桌面有一个文件名为`test.txt`的文本文件，通过"属性"命令我们可以查看到以下信息：



"位置"属性再加上文件名（`C:\Users\Subin\Desktop\test.txt`)就是这个文件的**绝对路径**。在Windows操作系统中，绝对路径就是从文件所在硬盘分区盘符开始（Linux系统从根目录，也就是`/`），一步步地进入文件夹直至文件名，就像以上帝视角观察文件所在位置。

如果我们打开Windows的命令提示符，默认定位至当前用户目录，如下图所示：



这样，`test.txt`文件的**相对**于用户目录的路径就是`Desktop\test.txt`，就像我们站在原地，看不到后面的路，只能向前寻找文件的路径。

步骤2 读取文件

无论是读文件还是写文件，对文件操作的第一步都是打开这个文件，之后对该文件进行后续的操作。

在file文件夹下有一个hello.txt的文本文件（你可以在当前目录下新建一个文件夹`file`，然后用记事本创建一个hello.txt，里面随便几行内容），我们尝试打开这个文件。

```
f1 = open('file/hello.txt','r')
```

`open()`函数用于通过指定的方式打开一个文件，并返回一个文件对象供后续操作使用。它需要两个参数：

> 文件路径：可以为绝对路径也可以为相对路径，建议为相对路径。
>
> 打开模式：常用的包括只读`r`，只写`w`，追加`a`；若文件为二进制文件，后接`b`即可（例如`rb`，`wb`，`ab`）。

由于我们现在只需要读取这个文本文件，所以文件打开方式为`r`。现在我们看看这个文件中有什么内容。

```
s = f1.read()
print(s)
```

我们使用了`read()`方法读取文件中的全部内容，该方法返回一个文件内所有内容的字符串。

不过请注意`read()`方法**不适合直接读取很大的文件**，可能会导致内存溢出。当然Python还提供了`readline()`和`readlines()`方法分别用于读取文件的一行内容或者读取文件的所有行，并将每一行字符串作为一个元素放入至一个列表中返回。

请注意：`readline()`和`readlines()`读取文件中的一行时也会读取行末的换行符（虽然我们用记事本打开的时候看不到），在后续处理时，可能需要使用到字符串的`strip()`方法去除掉换行符。

更方便的是，Python中可以利用for循环依次读取文件的每一行内容，例如我们现在想一行行地输出某个文件的内容：

```
f1 = open('file/hello.txt','r')
for line in f1:
    line = line.strip()  # 去除行末地换行符，因为下一行print函数默认结尾输出一个换行符，如果不去除的话会输出两个换行符（行末+print默认输出的换行符），造成多余空行现象。
    print(line)  # 如果不使用strip()方法，也可以设置print函数的默认输出为空字符串
print(line,end='')
```

步骤3 文件指针

如果我们再运行一次`f1.read()`命令，会发现什么也没有输出。

这是因为文件对象中暗含一个指针，当以`r`模式或者`w`模式打开文件时，文件指针指向文件的起始位置，当调用一次`read()`方法后读取文件全部内容，指针指向文件末尾，再次调用时因为指针已到达文件尾部，只会返回一个空字符串。

同样，当调用一次`readline()`方法时，文件指针会移动至下一行内容的开头。

我们可以利用`seek()`方法改变当前指针位置。

```
f1.seek(0)  # 将文件指针移动至文件起始点
s = f1.readline()
print(s,end='')  # 需要注意的是，print方法默认在最后输出一个换行符，由于readline方法读取的一行最后已经含有一个换行符，我们在print方法中需要指定输出后末尾再输出空字符串而不是默认的换行符。
```

如果以二进制模式打开文件，还可以从当前文件位置向后移动x个字节、从文件末尾位置向前移动x个字节。

最后还需要说明的是，从文件中读取的**全部都是字符串**，如果文件中包含数字信息，需要后续操作中使用`int()`方法或者`float()`方法进行类型转换。

步骤4 写入文件

现在我们尝试一下把数据写入至一个文件中。

假设我们有一个字典数据类型，里面是某日全国各地的平均气温，我们想将城市及其气温写入至一个txt文件中，格式例如：

```
北京 27.2
天津 26.5
上海 28.2
重庆 30.1
```

那么就需要利用`write()`方法写入文件，首先我们要创建一个新的txt文件，以`w`模式打开。

- **注意：** 如果文件不存在，以`w`模式打开文件会创建该文件；如果文件已存在，则会全部覆盖当前文件！

```
weather = {'北京':27.2, '天津':26.5, '上海':28.2, '重庆':30.1}
f2 = open('file/weather.txt','w')
```

`write()`方法的参数是待写入文件的**字符串**，由于我们的数据中有浮点类型，所以我们在写入之前需要将其转换为字符串类型的数据。

我们的思路可以是这样：

1. 读取字典中的每一项键值对;
2. 将值转换为字符串，并与键进行拼接，最后再拼接一个换行符;
3. 利用`write()`方法写入文件中。

```
for key,value in weather.items():
    s = key + ' ' + str(value) + '\n'
    f2.write(s)
```

现在我们从文件管理器中打开这个txt文件，看一下内容是否符合我们的预期。

步骤5 关闭文件

在对文件对象的所有操作完毕后，别忘了关闭文件对象释放资源。

```
f1.close()
f2.close()
```

当处理一个文件对象时, 使用 `with` 关键字是非常好的方式。在结束后, 它会帮你自动关闭文件。

```
with open('file/hello.txt','r') as f1:
    ...
```

实验6-3 Linux基本操作（上）

此部分内容可登录数据学院-功夫编程平台进行操作。

选择Terminal - Python3环境

Linux 英文解释为 ***L***inux ***i***s ***n***ot ***U***ni***x***

目前Linux已经被移植到更多的计算机硬件平台，远远超出其他任何操作系统。Linux可以运行在服务器和其他大型平台之上，如大型计算机和超级计算机。世界上500个最快的超级计算机已100%运行Linux发行版或变种。

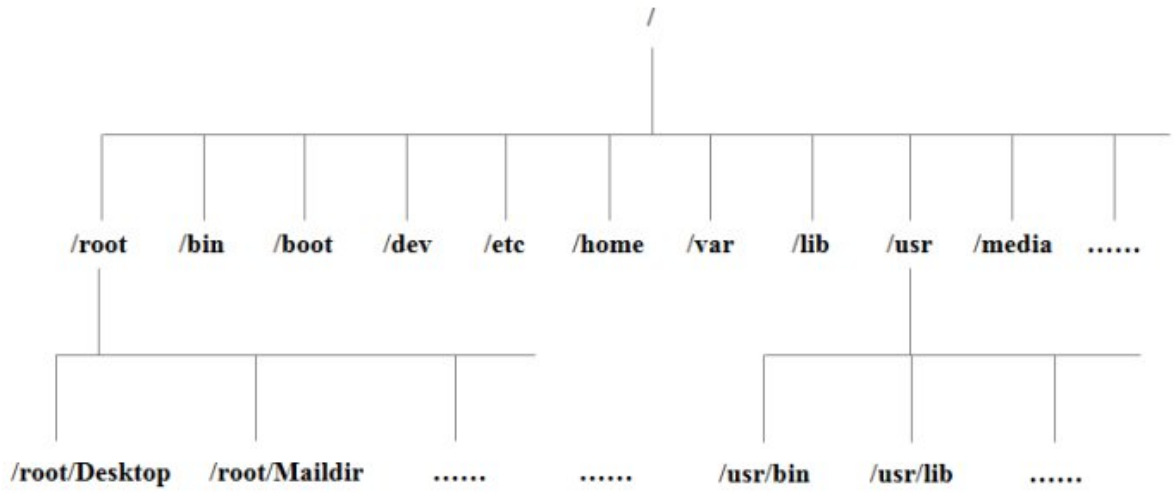
连Android手机操作系统都是基于Linux的！

我们来了解一下Linux命令行中常用的几个命令。

步骤1 `ls`命令

我们常用的Windows操作系统中，一块硬盘可能会分为多个分区，并配以C,D,E等等的“盘符”，前面我们学习绝对路径时也了解到一个文件是从盘符开始逐步定位至该文件。

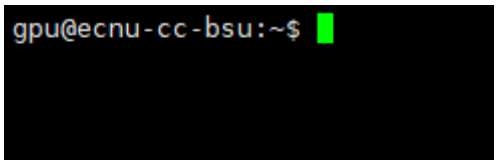
而Linux系统中文件目录结构就像是一棵倒立的树，一切都是从根目录`/`开始的，如下图所示。



我们在Windows中通过“文件资源管理器”利用鼠标可以方便地进入某个文件夹，查看目录下的所有文件。但是Linux系统更多的是命令行模式，用不到鼠标操作，一切都是通过键盘输入命令完成。

假设我们以gpu的用户名远程登录到一个Linux主机时，默认位于`/home/gpu`目录下（这个目录有一个别名“家目录”`~`），查看当前目录下的文件，需要使用`ls`命令。

现在我们看一下“家目录”下有什么文件。



屏幕上的`gpu@ecnu-cc-bsu:~$`意思是当前登录用户是`gpu`，Linux的主机名为`ecnu-cc-bsu`，目前位于`~`目录（也就是`/home/gpu`），`$`代表当前用户为普通用户。

我们输入`ls`后敲击回车。

```
gpu@ecnu-cc-bsu:~$ ls
anaconda3  bsu  examples.desktop  login.py  公共的  模板  视频  图片  文档  下载  音乐  桌面
gpu@ecnu-cc-bsu:~$
```

我们看到当前目录下的文件和文件夹，其中文件以白色表示、文件夹以粗体蓝色表示。

不过目录下文件数量不多还好，如果目录中文件过多（比如`/etc`），那么直接输入`ls`后可能会这样……

```
gpu@ecnu-cc-bsu:/etc$ ls
acpi                        geoclue             machine-id              rmt
adduser.conf                ghostscript         magic                   rpc
alternatives                glvnd               magic.mime              rsyslog.conf
anacrontab                  gnome               mailcap                 rsyslog.d
apg.conf                    gnome-system-tools  mailcap.order           sane.d
apm                         gnome-vfs-2.0       manpath.config          securetty
apparmor                    groff               mime.types              security
apparmor.d                  group               mke2fs.conf             selinux
apport                      group-              modprobe.d              sensors3.conf
appstream.conf              grub.d              modules                 sensors.d
apt                         gshadow             modules-load.d          services
avahi                       gshadow-            mtab                    sgml
bash.bashrc                 gss                 mtools.conf             shadow
bash_completion             gtk-2.0             mysql                   shadow-
bash_completion.d           gtk-3.0             nanorc                  shells
bindresvport.blacklist      hddtemp.db          netplan                 skel
binfmt.d                    hdparm.conf         network                 sound
bluetooth                   host.conf           networkd-dispatcher     speech-dispatcher
bonobo-activation           hostname            NetworkManager          ssh
brlapi.key                  hosts               networks                ssl
brltty                      hosts.allow         newt                    subgid
brltty.conf                 hosts.deny          nsswitch.conf           subgid
```

这简直就是灾难片现场！幸好我们可以在`ls`命令后附上`-l`参数，让文件和文件夹以列表形式呈现。

```
gpu@ecnu-cc-bsu:/etc$ ls -l
总用量 1192
drwxr-xr-x  3 root root    4096 7月   25  2018 acpi
-rw-r--r--  1 root root    3028 7月   25  2018 adduser.conf
drwxr-xr-x  2 root root    4096 9月    1 16:05 alternatives
-rw-r--r--  1 root root     401 5月   30  2017 anacrontab
-rw-r--r--  1 root root     433 10月   2  2017 apg.conf
drwxr-xr-x  6 root root    4096 7月   25  2018 apm
drwxr-xr-x  3 root root    4096 8月   17 16:27 apparmor
drwxr-xr-x  8 root root    4096 8月   31 16:29 apparmor.d
drwxr-xr-x  5 root root    4096 8月   31 14:15 apport
-rw-r--r--  1 root root     769 4月    4  2018 appstream.conf
drwxr-xr-x  7 root root    4096 8月   17 16:24 apt
```

确实是按列表方式显示文件和文件夹了，不过前面一大长串是什么东西呢？我们依次分析一下。

我们以第1行acpi的文件夹作为示例，可分为7个字段：

- 第1字段：文件/目录属性

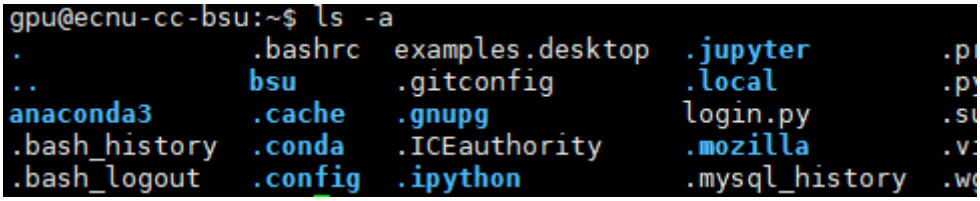
> drwxr-xr-x 共10位。第1位如果是 d 则代表这是一个目录，如果是 - 则代表这是一个文件，还有其他的字符我们这里暂不讨论；第2-4位代表文件拥有者的权限，r 代表读权限、w 代表写权限、x 代表执行权限，如果某位字母是 - 则代表该用户没有相应位的权限；第5-7位代表文件所有者的同组用户的权限；第8-10位代表其他用户的权限。

- 第2字段：硬链接数（这里暂不讨论）
- 第3字段：文件/目录拥有者
- 第4字段：文件/目录拥有者所在用户组
- 第5字段：文件/目录所占空间（默认单位为字节，可使用`ls -lh`参数显示以KB,MB或GB为单位的文件大小）

> 需要注意的是，**目录也是一个文件**，通过`ls -l`查看到的目录所占空间只是这个**目录文件**所占的空间大小，并不是子目录所有文件所占的空间，这与我们使用Windows系统的观念截然不同。

- 第6字段：最近修改时间
- 第7字段：文件/目录名

我们在Windows操作系统中知道有“隐藏文件”的概念，同样Linux也有隐藏文件的概念，使用`ls`命令默认不显示隐藏文件，如果需要显示隐藏文件，需要添加`-a`参数。



我们看到隐藏文件的文件名都是以`.`开头的。

当然上述所介绍的参数可以混合使用，且对参数顺序不敏感。

步骤2 `pwd`命令

`pwd`命令用于显示当前所在的目录，以绝对路径显示。在安装软件等场合下确认当前目录很有必要。

步骤3 `cd`命令

`cd`命令用于进入某一目录，参数为路径（绝对路径或相对路径）。

例如我们通过`pwd`命令得知目前位于`/home/gpu`目录，现在进入`/home/gpu/bsu`目录，通过输入绝对路径或相对路径可以选择如下命令之一：

> `cd /home/gpu/bsu`
>
> `cd bsu`

- 小技巧：如果目录名过长，可以先输入前几个字符，然后按下键盘的`TAB`键自动补全。

如果想进入父级目录(返回上一级目录），可使用`cd ..`

> `.` 代表当前目录
>
> `..` 代表父目录
>
> `../..` 代表父目录的父目录，以此类推

步骤4 `mkdir`命令

`mkdir`命令用于新建目录，如果创建多级目录，需要添加`-p`参数，例如：

> `mkdir test1`
>
> `mkdir -p test2/test3`

步骤5 `rmdir`命令

`rmdir`命令用于删除目录，如果删除多级目录，需要添加`-p`参数。

- 注意：待删除的目录内必须为空，否则需要使用`rm`命令（后文介绍）删除。

步骤6 `touch`命令

`touch`命令用于创建一个空文件，例如创建一个`a.txt`空文件：

```
touch a.txt
```

步骤7 `cp`命令

`cp`命令就是英文copy的简写，用于复制文件或目录，必需参数为源文件路径和目标文件路径。

例如我们目前在`/home`目录下，将`/home/gpu/s1.txt`复制到`/home/gpu/test1`目录下，并重命名为`s2.txt`

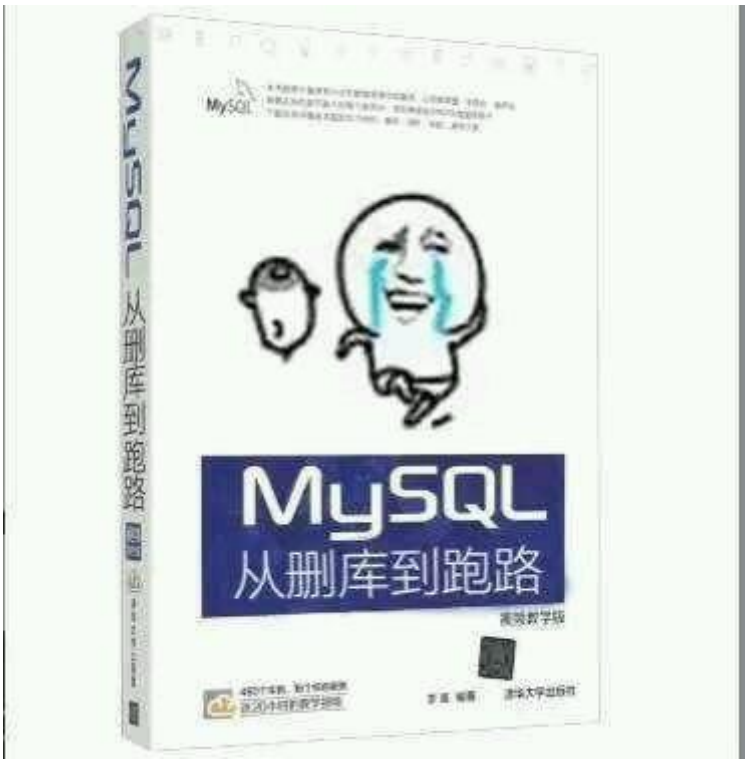
```
cp gpu/s1.txt gpu/test1/s2.txt
```

步骤8 `mv`命令

`mv`命令就是英文move的简写，用于移动文件或目录，也可以用于重命名文件或目录，必需参数与`cp`命令相同

步骤9 `rm`命令

`rm`命令就是英文remove的简写，用户删除文件或目录。Linux系统可没有什么“回收站”，所以**进行删除操作之前一定要三思**，否则很可能上演从删库到跑路的悲剧了……



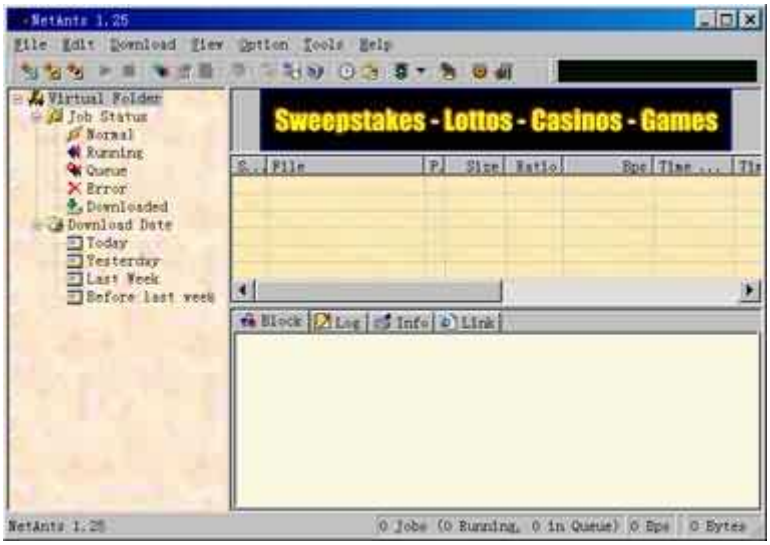
如果删除一个文件，可以直接使用`rm`后接文件名；

如果删除一个目录，需要加上`-r`选项，后接目录名；

`-f`选项将忽略不存在的文件，不会出现警告信息。

实验6-4 Python多线程初步（选做）

2000年左右，互联网在内地还处于初始发展阶段，家庭用户大都以ADSL方式拨号上网。下载速度最高也就400多KB每秒（在今天可以通过某度云下载文件重温当年的下载速率）。当时IE浏览器使用的是单线程下载，不过网络上也出现了很多下载工具，例如快车、网络蚂蚁等等，这些软件通过把文件分成多段，并发下载以提高下载速度。



这就好比一群蚂蚁同时搬运物资一样，单位时间可以得到多倍的资源。

在Python中有一个threading的库，可以创建线程类并实例化线程对象运行。现在我们写一个运行两个线程进行报时的小程序。

```
import threading
import time


class clockThread(threading.Thread):
    def __init__(self,threadID,name,count,delay_time):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.count = count
        self.delay_time = delay_time

    def run(self):
        print('线程编号：' + str(self.threadID) + '，开始运行！')
        for i in range(self.count):
            print(self.name + '为您报时：' + time.asctime())
            time.sleep(self.delay_time)
        print('线程编号：' + str(self.threadID) + '，运行结束！')
```

```
th1 = clockThread(8848,'松',10,1)
th2 = clockThread(8849,'竹',10,1)
th3 = clockThread(8850,'梅',10,1)

th1.start()
th2.start()
th3.start()
```

我们创建`clockThread`自定义类是继承自`Thread`父类，并且重写了`__init__`和`run`方法。其中类中两个重要的属性`threadID`和`name`分别代表线程编号和名称。程序开始运行时，各线程开始竞争CPU资源。

当然多线程需要考虑许多问题，例如线程同步、优先级队列等等问题，这些会在《操作系统》课程中会有详细的探讨。

实验练习06

1. 实验6-2的步骤2中介绍了读取整个文件的`read()`方法，请自己用记事本创建一个txt文件，里面写上几行内容，再编写程序，体会`readline()`和`readlines()`方法的作用。
2. stuGrade.csv文件中包含5位同学的学号、语文成绩、数学成绩和英语成绩，请先用记事本打开该文件，查看文件内容，然后编写Python程序，读取该文件，并计算出各科的平均成绩（保留2位小数）。
3. 请在第2题的代码继续编写Python程序，将以下内容写入my.txt文件中：

> 第1行：你的学号和姓名
>
> 第2行：3门课程的平均成绩，保留2位小数，以英文逗号间隔
>
> 第3行：系统当前时间，格式例如2020-01-01 12:00:00
>
> 第4行：两秒后的系统当前时间（利用`sleep()`方法）

4. 在数据学院-功夫编程平台练习实验6-3的相关命令