**第9章 文 件**

计算机中的文件(File )是以硬盘等外部介质为载体，存储在计算机中的信息的集合，如 文本文档、图片、程序等都是文件。文件在计算机中应用广泛，本章将对Python中文件的相关 概念，以及在Python中与文件操作相关的方法进行讲解。

9O1

在Windows系统中，文件被具象为一个个图标，若想修改文件中的内容，需要先双击打开 文件；若文件使用完毕，需关闭文件。在程序中操作文件的步骤与通过图形界面操作文件的步 骤类似,同样离不开文件的打开和关闭。

9.1J打舞支繹

Python中可通过内置方法open()打开文件，open()方法的声明如下：

open (file, mode= 1 r ! f buf fering=-l)

该方法中的参数file 一般为文件的路径名；参数mode用于设置文件的打开模式，默认值 为“产，表示以只读模式打开文件；参数buffering用于设置访问文件时采用的缓冲方式，默 认值为表示采用系统默认的缓冲机制。若使用opm()方法成功打开文件，则返回一个文件 流，否则设置emioo

9J.2桑揚黨鱒

Python中可通过内置方法close()关闭文件。假设使用open()方法打开一个文件,并将文 件流赋给对象f,如下所示。

>>> f = open ( f a.txt!)

在文件使用完毕后通过close()方法关闭文件,具体代码如下:

»> f. close ()

程序执行完毕后，系统会关闭由该程序打开的文件，但更好的做法，是在文件使用完毕后， 由程序调用close()方法关闭文件。

实际上，计算机中可打开的文件数量是有限的，每打开一个文件歹就会占用一个“名额”膏 当该“名额”耗尽后，系统将无法再打开新的文件。此外，当文件以缓冲方式打开时，磁盘文 件与内存间的读写并不是即时的，若程序因异常关闭，可能导致缓冲区中的数据无法写入文件， 造成数据丢失。

综上所述，在文件使用完毕后，及时使用clo院()方法关闭文件是有必要的。

9e 2 :W慨述

9.2J躇篷畲

一个文件需要有唯一确定的文件标识，保证用户可根据标识找到唯一确定的文件。文件标 识包含三个部分，分别为文件路径、文件名主干、文件扩展名。如图94为一个文件的完整标识， 根据该标识可以找到D:\itcast\chapterlO路径下扩展名为Jat9文件名为example的二进制文件。

D:\itcast\chapterlO\example. dat

路径 文件名主干文件扩展名

图94文件标识

若要打开的文件存放在当前工作目录，则可省略文件的路径。示例如下：

>>> f = open(\* a o txt1)

执行如上语句，将会打开当前路径下的&txt文件，并将打开的文件流赋给文件对象f。

凱22氣舞膜饑 ’

opoi()方法中的mode参数用来设置文件的打开模式，该参数通常是一个字符，包括：r. w、 a. b、+,这些字符各自代表的含义分别如下：

* r：以只读的方式打开文件。
* w：以只写的方式打开文件。
* a：以追加的方式打开文件。
* b：以二进制形式打开文件。
* +：以更新的方式打开文件(可读可写)。

若要以只写模式打开当前路径下的文件alxt,可使用如下语句：

>>> f = open(\* a »txt? !w \*)

需要说明的是，用于设置文件打开模式的字符可搭配使用，常用的文件打开模式如表9J 所示。

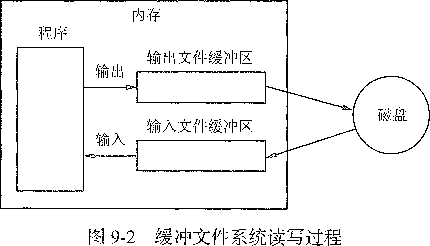
表9』文件打开模式

|  |  |  |
| --- | --- | --- |
| 打开模式 | 名 称 | 描 述 |
| r/rb | 只读模式 | 以只读的形式打开文本文件/二进制文件，如果文件不存在或无法找到， open（）函数将调用失败 |
| w/wb | 只写模式 | 以只写的形式打开文本文件/二进制文件，如果文件已存在，重写文件?若文  件不存在则创建文件 |
| a/ab | 追加模式 | 以只写的形式打开文本文件/二进制文件，只允许在该文件末尾追加数据，如 果文件不存在，则创建新文件 |
| r+/rb+ | 读取（更新）模式 | 以读/写的形式打开文本文件/二进制文件，如果文件不存在,並即（）函数调用  失败 |
| w+/wb+ | 写入（更新）模式 | 以读/写的形式创建文本文件/二进制文件,如果文件已存在，则重写文件 |
| a+ab+ | 追加（更新）模式 | 以读/写的形式打开文本/二进制文件,但只允许在文件末尾添加数据,若文 件不存在，则创建新文件 |

其中“产是open。的默认打开方式，即若参数mode缺省，则文件会以只读模式打开。此 外需要说明的是，由于所有POSIX兼容的UNIX及类UNIX系统将文件都视为二进制文件，因 此对这些系统来说,%”是可有可无的，但为了保证程序的可移植性（防止在不同系统间移 植时岀错），在明确指定打开二进制文件时不应缺省％” O

在C语言中,文件系统分为缓冲文件系统（标准I/O）和非缓冲文件系统（系统I/O） o

若为缓冲文件系统，系统会在内存中为正在处理的程序开辟一段空间作为缓冲区，若需从磁盘

读取数据，内核一次将数据读到输入缓冲区 中，程序会先从缓冲区中读取数据，当缓冲 区为空时，内核才会再次访问磁盘；反之若 要向磁盘写入数据，内核也先将待输出数据 放入输出缓冲区中，待缓冲区存满后再将数 据一次性写入磁盘。缓冲文件系统中文件的 读写过程如图9.2所示。

而在非缓冲文件系统中，每次读写磁盘

中的数据，都要对磁盘进行访问。

Python中open（）方法的参数buffering可用来设置访问文件的缓冲方式，若buffering设置 为0,则表示采用非缓冲方式；若设置为1则表示每次缓冲一行数据；若设置为大于1的值则 表示使用给定值作为缓冲区的大小。当然若参数buffering缺省，或被设置为负值时，表示使用 默认缓冲机制（由设备类型决定）o

相比内存与缓冲区之间的读写，内存与磁盘间的读写时间消耗更大，因此，采用有缓冲的 打开方式可减少内存与磁盘的交互次数，提高文件读写的效率。

凱2诚黨繹阚翁餐 .

计算机中的文件分为两类，一类为文本文件，另一类为二进制文件。

文本文件又称为ASCH文件，该文件中一个字符占用一个字节，存储单元中存放单个字 符对应的ASCII码。假设当前需要存储一个整数数据112185,则该数据在磁盘上存放的形式 如图9；所示。

'1' (49) T (49) '2' (50) T (49) 8 (56) 5 (53)

00110001 00110001 01010000 00110001 01010110 01010011

图9-3文本文件存放形式

由图9-3可知，文本文件中的每个字符都要占用一个字节的存储空间，并且在存储时需要 进行二进制和ASCII码之间的转换，因此使用这种方式既消耗空间，又浪费时间。

数据在内存中是以二进制形式存储的，如果不加转换地输出到外存，则输出文件就是一 个二进制文件。二进制文件就是存储在内存的数据的映像，也称为映像文件。若使用二进制 文件存储整数112185,则该数据首先被转换为 112185

二进制的整数，转换后的二进制形式的整数为 pK)oooo峋 oooooooi | louoiio [ooniooi 11011011000111001,.此时该数据在磁盘上存放的 囱n 4 一港判卄仆&汕

图9-4 一进制文件存放形式 形式如图9-4所示。

对比图9・4和图9・3可以发现，使用二进制文件存放时，需要的存储空间更少，并且不需 要进行转换，如此既节省时间，又节省空间。当然这种存放方法不够直观，需要经过转换后才 能看到存放的信息。

凱2忐瀛濮支俸 .

Python的sys模块中定义了 3个标准文件，分别为stdin (标准输入文件)、stdout (标准 输岀文件)和stderr (标准错误文件)，标准输入文件对应输入设备，如键盘；标准输出文件 和标准错误文件对应输出设备，如显示器。

在解释器中添加sys模块后，便可使用Python对象获取这三个对象，示例如下：

import sys

f = sys.stdout

>>> foWrite("hello")

hello5

以上代码将标准输出文件赋给文件对象f,又通过文件对象f调用内置方法write()向标准 输岀文件写数据。观察代码执行结果，“hello”被成功写到了标准输出中(h©血之后的5表示 本次写到标准输出中的数据的字节数)。

每个终端都有其对应的标准文件，这些文件在终端启动的同时打开。

I 9O 3支费的读写

在图形界面打开文件后，用户可直接通过键盘等输入设备，对文件进行编辑，但在程序中, 文件的读写操作还需使用相关的函数实现。

Python中可通过write()方法向文件中写入数据,write()方法的定义如下:

write(str)

其中毗表示要写入文件中的字符串。若调用成功则返回本次写入文件中的字节数。 假设使用open()方法打开一个文件，并将文件流赋给对象f,如下所示。

>>> f = open(1 a.txt \*)

则可以使用以下语句向文件alxt中写入数据:

>>> f.write(nhello itheima.\n°)

#本次写入文件的字节数

#本次写入文件的字节数

#本次写入文件的字节数

#关闭文件

*n* . . ■■ ■ "二；*」*…

3 he 1 t the i ms.

hello *Id.*

15

>>> f.write("hello itcast.")

13

>>> f.write("hello world.")

12

f.close ()

此时文件a.txt中的内容如图9・5所示。

由图9-5中的写入结果可知9 write。每次写 入后并不会主动换行，而是在上次写入的数据之 后继续写入数据，因此，若需向文件中写入一行 数据，应在字符串末尾添加换行符"\11” O

图9-5 write()写入结果

Python中读取文件内容的方法有很多,其中最常用的有read() readline()和readlines()。 1Q read()

readO方法可从指定文件中读取指定字节的数据,该方法的定义如下:

read(size)

其中参数size用于指定从文件中读取的数据的字节数，若参数size缺省，则一次读出指定 文件中的所有数据。

以读取文件a.txt中的数据为例，read()的用法如下。

»> f = open (" o / a o txt")

»> f .read(5) #从a . txt中读取5个字节

fhello?

»> f . read (3) #从a。txt中读取3个字节

1 if

»> f • read () #读取剩余的全部数据

\*heima.\nhello itcast.hello world.5

>» f. read.()

'' #再次调用，读取到的数据为空

>>> f.close ()

由以上示例可知，每次调用『ead()函数时，函数会从上次读取位置继续向下读取数据。

1. readiineO

readlineQ方法每次可从指定文件中读取一行数据，以读取文件凯txt中的数据为例, readlineQ的用法如下。

>>> f = open("./a.txt")

>>> f.readline()

* hello itheima. \n \* # 第一行数据

>>> f.readline()

* hello itcast. hello world. \* # 第二行数据

>» f. readline ()

>>> f.close()

1. readlinesO

readlinesQ方法可将指定文件中的数据一次读出,并将每一行视为一个元素,存储到列表 之中。以读取a.txt中的数据为例，readlinesQ的用法如下:

>>> f = open("a.txt")

*»> f*. readlines ()

[?hello itheima*.\n'f* 'hello

>>> s *= f*«readlines()

>>> type(s)

<class \* list \*>

>>> f.close()

itcast.hello world.1]

#将读取的数据存储到对象s中 #获取对象s的类型

# S的类型为列表

read()(参数缺省时)和readlines()都可一次读出文件中的全部数据，但这两种操作都不够 安全。因为计算机的内存是有限的，若文件较大，貝d()和readlinesQ的一次读取便会耗尽系统 内存，这显然是不可取的。当然"若文件中存在数据量较大的行，readlineQ同样不够安全。

经过931和932小节的学习我们发现，文件在一次打开与关闭之间进行的读写操作都是 连续的，程序总是从上次读写的位置继续向下进行读写操作。实际上，每个文件对象都有一个 称为“文件读写位置”的属性，该属性用于记录文件当前读写的位置。

Python中提供了一些获取文件读写位置以及操作文件读写位置的方法，下面将对这些方法 进行讲解。

t tell()

用户可通过伯1()方法获取文件当前的读写位置。以操作文件飢txf为例，@1()的用法如下 所示。

>>> f = open("a.txt")

>>> f.tell()

0

>>> f.read(5)

fhello\*

>>> f.tell ()

5

>>> f.close()

由以上示例可知，打开一个文件后，文件默认的读写位置为0;当对文件进行读操作后， 文件的读写位置也随之移动。

2O seek()

一般情况下，文件的读写是顺序的，但并非每次读写都需从当前位置开始。Python »了 院ek()方法，使用该方法可控制文件的读写位置，实现文件的随机读写。s炎k()方法的声明如下：

seek(offset,from)

其中参数offset表示偏移量，即读写位置需要移动的字节from用于指定文件的读写位置, 该参数的取值为：0、1、2,它们代表的含义分别如下：

® 0：表示文件开头。

*。*1：表示使用当前读写位置。

。2：表示文件末尾。

seek()调用成功会返回当前读写位置。

以操作文件alxt为例，seek()的用法如下所示。

>>> f = open("a «txt")

>>> f•tell ()

0

»> f.seek(5,0) #相对文件开头进行偏移

5

需要注意的是，在Python 3中，若打开的是文本文件，那么seek()只允许相对于文件开头 移动文件位置，即若在参数fix)m值为1、2的情况下对文本文件进行位移操作，将会产生如下 的错误。

»> f. seek (4,1) #相对当前读写位置进行偏移

Traceback (most recent call last):

File "<stdin>"f line 17 in <module>

io.UnsupportedOpeisation: can v t do nonzero cur-relative seeks

换言之，若要相对当前读写位置或文件末尾进行位移操作，需以二进制形式打开文件，示 例如下：

»> f = open (n a. txt °/nrb °) #以二进制文件形式打开

»> f o seek (5,0) #相对文件开头进行偏移

5

»> f.seek(4,l) #相对当前位置进行偏移

9

»> f .seek(5f 2) #相对文件末尾进行偏移

45

»> f„seek(-3,2) #相对文件末尾进行偏移

37

>>> f.close ()

9.4曳件的处渗U属性

除open。、close。、write()和read。系列方法之外，文件还有一些相关的内置方法和属性， 这些方法和属性的名称及功能分别如表头2所示。

表9-2文件常用方法和属性

|  |  |  |
| --- | --- | --- |
| 分 类 | 名 称 | 功能 |
| 方法 | fileno() | 返回底层文件的文件描述符 |
| flush() | 清空文件对象，将缓存中的内容写入磁盘 |
| readableQ | 若文件对象已打开且等待读取，则返回True,否则返回False |
| seekable() | 若文件支持随机存取，则返回True,否则返回False |
| truncate(size) | 截取文件到当前文件读写位置，若给定size,则截取为size大小 |
| \_next\_() | 返回文件对象的下一行 |
| 属性 | mode | 获取文件对象的打开模式 |
| name | 获取文件对象的文件名 |
| encoding | 获取文件使用的编码格式 |
| closed | 若文件已关闭则返回True,否则返回False |

9o5曳滞遍鹰 「

文件遍历的实质是对文件的读取。Python中可通过read()、readlineQ和readlines()方法获 取文件内容，这些方法通常与循环搭配使用，实现文件内容的打印。下面将结合案例，分别对 这些方法进行讲解。

read()方法可以从指定文件中读取指定字节的内容，其用法如下:

file o read ( [size])

以上示例咬d()方法中的参数血。用于设置本次读取的字节数，该参数可以省略，若省略, 则冀ad()方法将会一次性读取指定文件中的全部内容。

下面通过一个案例来展示冀ad()的用法，在该案例中使用『sd()函数遍历文件，每次获取 固定字节的数据，并在循环中打印文件内容。

案例代码如下：

fileName = input (叩-清输入文件名：°) f = open (fileName)

s = f.read(10)

while s:

print(s fend="°)

s = f.read(10)

f.close ()

需要注意的是，print()函数默认情况下输岀数据后会自动换行，为保证打印出的文本格式 与文件内容格式一致，应将print()函数中end的值设置为f (空字符串)。此外，文件的读写

位置随着read()的调用每次向后偏移10个字节，直到文件末尾。

执行程序，假设输入的源文件名为&fxt,该 文件处于程序所在路径下，其中的内容如图9・6 所示。

则程序的执行结果如下所示。

*「""關「一題氫*

*饗關5編5?"編5"關蠶*

hsl k? i d hel Io，itcsst h仕il。 i theifTK

图 9-6 a.txt

请输入文件名:a. txt

hello world

hello itcast

hello itheima

®o5n2 r®arfin@0 遢癇1 密繹

readline()方法的用法如下:

file . readline ( [size])

readline()方法默认每次从指定文件中读取一行内容，但若为其设置参数size,则会从当前 行读取size字节的字符。

下面通过一个案例来展示readline()的用法,该案例使用readline()函数遍历文件,并打印 文件内容。

示例如下:

fileName = input ("请输入文件名：° )

f = open (fileName)

s = f.readline()

while s:

print(s,end=n") s = f•readline()

f.close ()

执行程序，程序执行结果如下所示。

请输入文件名:a. txt

hello world

hello itcast

hello itheima

在使用readline()函数遍历文件时，文件的读写位置随着readlineQ函数的调用不断移动， 每次向下偏移一行。

9o5h3 readlinesO 遍癒安牆

与read。方法类似,readlinesQ方法也可一次性读取指定文件中的全部内容，但不同的 是，readlinesQ方法会将读取到的内容存储到列表中，列表中的每个元素对应文件中的每一行。 readlines()的用法如下所示。

file . readlines ()

下面通过一个案例来展示readlines()的用法，该案例使用readlinesQ函数遍历文件，打印 文件内容。案例代码如下：

fileName = input (° 请输入文件名:")

f = open (fileName)

s = f.readlines()

for line in s:

print(linef end=°")

f.close()

执行程序，程序执行结果如下所示。

请输入文件名：a. txt

hello world

hello itcast

hello itheima

由readlinesQ函数一次便读取文件中的全部内容，因此在一次readlinesQ函数调用后， 文件的读写位置偏移到了文件末尾。

凱鼠凰黨鮮選備器

迭代是一个过程的多次重复，在Pythg中，实现了 \_\_iter\_()方法的对象都是可迭代对象(如 序列、字典等)。文件对象也是一个可迭代对象，这意味着可以在循环中通过文件对象自身遍 历文件内容。

下面通过文件对象实现文件的遍历与打印。示例如下：

fileName = input ("请输入文件名：°)

f = open (fileName)

for line in f:

print(line,end="")

f.close ()

执行程序，程序执行结果如下所示。

请输入文件名:a o txt

hello world

hello itcast

hello itheima

9.5.5黨IWft器齣顧慮

迭代器有“记忆"功能，若在第一次循环中只打印了部分文件内容，后续再次通过循环获 取文件内容时，会从上次获取到的文件内容后开始打印数据。

下面通过一个案例来验证迭代器的这个特点，案例要求为：使用迭代器获取文件内容，在 循环中打印文件，在两个有先后次序关系的循环中各打印2行数据。

示例如下：

fileName = input (” 请输入文件名:n)

f = open (fileName)

print (° 第一次扌丁E卩 °)

i = 1

for line in f:

print(linez end=°")

i += 1 if i==3:

break

print (" 第二次打印 研)

i = 1

for line in f:

print(line,end="n)

i += 1 if i==3:

break

f o close()

执行程序，假设输入的文件名为翊投xt,该文件处于程序所在路径下，其中的内容如

图9・7所示。

|  |  |  |
| --- | --- | --- |
| 则程序的执行结果如下: | *蟹懿蝠讒廳讒点矚鑫餾聽讒讒矚讒讒瀛覊聽編養*  ?祐緇推.滅席緇;:醐,做液細混緋捋耕辎絲辎辭输角細辭辎繚螂?.雜髄維男螂發湖勰法総細您紛雜卷 *饕爨囊囂翊蘿潔矚翌襪鬟囊"鑼?"餐餐繼鬟豪雾饗巒M* | *膏畫?f養養w言？\** |
| 请输入文件名:test. txt  ——第一次打印—— | 1. 郭况屁 2. 和T吕 3. itoast   *4* |  |
| 1 hello | 1. r i sb 2. -python |  |
| 9 wc r〕rS *i* |  |
| 乙 VVA\_> J\_ J\_  第二次打印 --- | 图 9-7 test.txt |  |

1. itcast
2. itheima

文件对象创建成功后便可获取文件中的内容，虽然readlinesO与迭代器功能类似，在一次 调用后便能获取整个文件中的数据，但若将本小节案例中的文件对象f替换为接收f.readlinesQ 返回值的对象，程序的两次打印将会打印出相同的内容，即每次遍历readlines()函数的返回值时， 都会从头开始遍历。

9o6

文件备份是与文件相关的一项常用操作。实际上，文件备份也通过文件的打开和读写实现。 假设当前目录下有一个文件Mxt,要在当前目录下制作该文件的备份一一a［备份］Ixt,则具体

步骤如下所示。

1. 以只读的方式打开源文件alxto
2. 创建新文件F［备份"xt” *，*以只写的方式打开。
3. 读取源文件a.txt中的内容。

(4 )将文件alxt中的内容写入新文件a［备份］lxt中。

(5 )关闭文件a.txt和a［备份］.txto

下面通过一个案例来实现文件的备份操作，案例代码如下所示。

1

2

3

4

5

6

7

8

9

10

11

12

13

14

file\_name = input ("请输入要拷贝的文件名字:") source\_file = open (file\_namer 1 r 1 ) if

打开源文件

source\_file : flag = file\_name . rfind *('。)* if flag > 0:

file\_flag = file\_name [flag:] copy\_file = file\_name [ :flag]

#创建新文件\*以只读的方式打开 new\_file = open (copy\_\_filef J w,)

#逐行复制源文件内容到薪文件中

for line\_content in source\_file . readlines (): new\_file. write (line\_content) source\_\_file。close () new file . close ()

获取文件后缀起始位置 拼接新文件名

\* [备份]? + file\_flag

#关闭源文件

#关闭备份文件

以上代码中，第2行的open()方法用于创建新文件，第11行的readlines()方法用于获取 源文件中的内容，第12行的write。方法用于将源文件中内容逐行写入备份文件中。假设输入 的源文件名为程序执行成功后，会在源文件的路径中新增一个新文件％［备份Hxt” *，* 打开该文件，如果其中内容与中内容完全相同，说明文件备份成功。

9e7玄铮重命名 . .

Python中提供了更改文件名的方法 renameQ,该方法存在于os模块中，其格式如下:

rename (原文件名,新文件名)

若要使用该方法，需先在Python中加载os模块。该方法的用法如下所示。

>>> import os

>>> os.renmae("a.txtn,"test•txt")

经以上操作后，当前路径下的文件被重命名为test.txto需要注意的是，待重命名的文 件必须已存在，否则解释器会报错。

9O8 i•曳俳

Python中用于删除文件的方法为remove()?该方法同样存在于os模块中，其格式如下：

remove (文件名)

在解释器中调用该方法后，指定文件将会被删除，例如删除文件"testlxt",则可使用如 下语句：

>>> import os

»> os o remove ("test. txt")

调用remove()方法前同样要求目标文件已经存在。

I 0.9 os

除基本的删除和重命名操作外，OS模块还包含了一些与文件夹操作和目录操作相关的方法， 包括创建文件夹、删除文件夹、获取当前目录、更改默认目录、获取目录列表等。

凱凱創篷黨俸褰

os模块中的mkdirQ方法用于创建文件夹,其使用方法如下:

>>> import os

>>> os.mkdir("dir")

经以上操作后，解释器会在默认路径下创建文件夹di『。待创建的文件夹不能与已有文件 夹重名，否则将会创建失败。

9.9.2觀臘裒瓣褰

OS模块中的mdir()方法用于删除文件夹，其使用方法如下:

»> import os

>>> os o rmdir(°dirn)

经以上操作后，默认路径下的文件夹dif将被删除。若待删除的文件夹不存在将会报错。

当前目录即Python当前的工作路径。os模块中的getcwd()方法用于获取当前目录，其使 用方法如下:

>>> import os

>>> os • getcwd.()

5 C:\\Users\\admin \*

经以上操作后,当前目录“C:\\Users\\admin”被打印到终端。

気凱感•

os模块中的chdir()方法用来更改默认目录。若在对文件或文件夹进行操作时，传入的是 文件名而非路径名，Python解释器会从默认目录中查找指定文件，或将新建的文件放在默认目 录下。若没有特别设置，当前目录即为默认目录，如991节中创建的文件夹血便被放在当前 目录 “C:\\Users\\adminn 中。

若将默认目录更改为“E:\\”，那么新建的文件将会被放置到更新后的默认目录中。示例如下:

>>> import os

>>> os.chdir(nE:\\n)

>>> os.mkdir("dir°)

查看路径“E:\\”下的文件，找到新建的目录di「说明默认目录更改成功。

在实际应用中，经常会先获取到指定目录下的所有文件，再对目标文件进行相应操作。

Python的os模块中提供了 listdir()方法，使用该方法可方便快捷地获取一个存储指定目录下所 有文件名的元组。示例如下：

>>> import os

»> lis = os . listdir ("./")

经以上操作后，当前目录下的所有文件名都被存储到元组lis中，用户可通过遍历元组歹 获取目录中的文件。

I 9ol0曳停操作案例——關户登录

随着智能设备的普及和网络的不断发展，各种App如雨后春笋，层出不穷。在这些App中， 用户登录是其最基本的模块，本节将以登录模块为主，展示文件在程序开发中的用法。

0J0J摧麝還醴 ’’

本节要实现的用户登录功模块分为管理员登录和普通用户登录，详细业务逻辑如图9・8 所示。

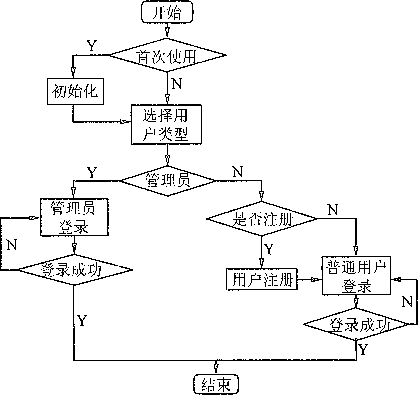


图9-8用户登录模块业务逻辑

从图9・8可知，在用户使用软件时，系统会先判断用户是否为首次使用：若是首次使用° 则进行初始化，否则进入用户类型选择。用户类型分为两种：若选择管理员，则直接进行登录;

若选择普通用户，则询问用户是否需要注册。若需要注册，则先注册用户呀再进行登录。

'0J0.2帼H莫俸

用户管理模块包含的文件有：

(1 )标识位文件flago

(2 )管理员账户文件u\_n)ot。

(3)普通用户账户文件。

标识位文件用于检测是否为初次使用系统，其中的初始数据为0,在首次启动系统后将其 数据修改为1;管理员账户文件用于保存管理员的账户信息，该账户在程序中设置，管理员账 户唯一；普通用户文件用于保存普通用户注册的账户，每个用户对应一个账户文件，普通用户 账户被统一存储于普通用户文件夹users中。

凱10总滇皴购饑 ’

用户登录模块应包含的函数及其功能分别如下：

1. is\_first\_start():判断是否首次使用系统;

(2 ) c\_flag():标识位文件更改;

(3)init()：信息初始化；

(4 ) print\_login\_menu():打印登录菜单;

(5 ) user\_select():用户选择;

(6 ) root\_login():管理员登录；

(7 ) user\_register():用户注册；

(8 ) user\_login():普通用户登录。

9J0.4功罷褰嬲

1 B\_fi rst\_start()

is\_first\_start()函数首先用于判断是否为首次使用系统，为保证每次读取到的都为同一个标 志位对象，我们需要一个标志位文件fhg,将标志位对象的初始值0存储到文件中。每次启动 系统后调用is项Tst\_start()函数打开Hag文件，从其中读取数据，并进行判断。

此外is\_fkst\_start()函数还将根据标识文件的判断结果执行不同的分支：若是首次启动，则 更改标志位文件内容、初始化资源、打印登录菜单，之后进行用户选择；若不是首次启动，则 直接打印登录菜单，并接收用户选择。.

is\_first\_start()函数的实现如下:

#判断是否为首次使用系统 def is\_first\_start (): flag = open ("flag") word = flag o read () if word == °0n:

print ("首次启动! ° )

flag. close () # 关闭文件

c\_flag ()

#更改标志为1

#初始化资源

#打印登录菜单

#选择用户

inito print\_login\_menu() user\_select() elif word == "1":

print ("欢迎回来!

print\_\_login\_menu () user\_select()

else :

print ("初始化参数错误!")

2. cjlag()

c\_flag()函数用于修改flag文件中的内容，将在初次启动系统时被is\_first\_start()函数调用。 该函数的实现如下：’

#更改标志位 def c\_flag ():

f = open (nflagn, ??wn) f.write (nln)

f.close ()

#以重写的方式打开文件flag

#将写入falg文件中

#关闭文件

3b init()

初次启动系统时，需要创建管理员账户和普通用户文件夹，这两个功能都在init()函数中 完成。inito函数的实现如下:

#初始化管理员用户

def init():

file = open ( "u\_root" r nw" ) #创建并打开管理员账户文件

root = {"rnum":"root"f "rpwd":"123456"}

|  |  |
| --- | --- |
| file . write (str (root)) | #写入管理员信息 |
| file o close () | #关闭管理员账户文件 |
| os.mkdir("users") | #创建普通用户文件夹 |

4a printjogin\_menu()

print\_login\_menu()函数用于打印登录菜单，该菜单中有两个选项，分别为管理员登录和普

通用户登录。print\_login\_menu()函数的实现如下:

#打印登录菜单

def print\_login\_menu(): print -用户选择- print (" 1-管理员登录”) print ("2-普通用户登录") print (° ——

5. user\_select()

在打印出登录菜单后，系统应能根据用户输入，选择执行不同的流程。此功能在 selecto函数中实现，该函数首先接收用户的输入，若用户输入 T , 调用rootjogin()函数

进行管理员登录；若用户输入“2”,则先询问用户是否需要注册。user\_select()函数的实现如下：

#用户选择

def user\_\_se].ect ():

while True:

user\_type\_select = input (° 请选择用户类型:")

if user\_type\_select == ° 1" : # 管理员登录验证

root\_login()

break

elif user\_type\_select == "2° : # 普通用户

while True:

select = input ("是否需要注册? (y/n):")

if select == or select == °Yn:

print ("一一用户注册 °)

user\_register () # 用户注册

break

elif select == "n" or select == nN°: print (" 用户登录 时)

break

else :

print ("输入有误,请重新选择°) user\_login () # 用户登录

break

else :

print (°输入有误,请重新选择°)

& rootJoginO

root\_login()函数用于实现管理员登录，该函数可接收用户输入的账户和密码，将接收到的 数据与存储在管理员账户文件u\_root中管理员账户信息进行匹配，若匹配成功则提示登录成功, 并打印管理员功能菜单；若匹配失败则给出提示信息，并重新验证。

root\_login()函数的实现如下:

#管理员登录

def root\_login():

while True:

print (”\*\*\*大 管理员登录 \*\*\*\*•) root\_number = input (° 请输入账户 名:° ) root\_password = input ("请输入密码:") file\_root = open (°u\_root°) # 只读打开 root 账户文件

root = eval (file\_root» read () ) # 读取账户信息

#信息匹配

if root\_number == root["rnum"] and root\_password. == root["rpwd"]: print尸登录成功！")

break

else :

print (°验证失败I ")

*7.* user\_register()

user\_\_register()函数用于注册普通用户。当用户在user\_select()函数中选择需要注册用户之 后，该函数被调用。user\_sekct()函数可接收用户输入的账户名、密码和昵称，并将这些信息保存到users文件夹中与账户名同名的文件中。

user\_register()函数的实现如下:

#用户注册

def user\_register():

user\_id = input (”请输入账户名:") user\_pwd = input ("请输入密码：° ) user\_name = input ("请输入昵称:°)

user = {nu\_id°:user\_idz °u\_pwd":user\_pwdz"u\_name°:user user\_path = "./users/" + user id

name}

file\_user = open (user\_pathA nw°)

#创建用户文件

#写入

#保存关闭

file\_user. write (str (user)) file\_user。close ()

& userjogin()

user Jogin ()函数用于实现普通用户登录，该函数可接收用户输入的账户名和密码，并将账 户名与users目录中文件列表的文件名匹配，若匹配成功，说明用户存在，进一步匹配用户密码, 账户名和密码都匹配成功则提示“登录成功/，并打印用户功能菜单。若账户名不能与us\* 目录中文件列表的文件名匹配，则说明用户不存在。

user\_login()函数的实现如下:

#普通用户登录

def user\_login():

while True:

print ("\*\*\*\*普通用户登录\*\*\*\*") user\_id = input (“请输入账户名：°) user\_pwd = input (叫请输入密码：°) # Mauser目录中所有的文件名

user\_\_list = os . listdir (" . /users")

#遍历元组？判断user\_id是否在元组中 flag = 0

for user in user\_list:

if user == user\_id.:

flag = 1

print (n 登录中

#打开文件

file\_name = " . /users/"+user\_id file\_user = open (file\_name)

#获取文件内容

user\_inf o = eval (file\_user . read ()) if user\_pwd == user\_info["u\_pwd"]: print ( °登录成功！") break

if flag == *1:*

break

elif flag == 0:

print ("查无此人!请先注册用户° )

break

凱10忐»8瀛泰

到此，用户登录模块所需的功能已全部实现。需要注意的是，由于初始化函数诅计()和用 户登录函数user\_login()中使用了 os模块的listdirQ方法，所以需在程序开头添加。s模块，如 下所示。

import os

此后在文件末尾添加如下代码，便可开始执行程序言

is\_first\_start ()

以上代码用于调用■is\_first\_start()函数，该函数是用户登录模块的核心功能，也是其他各函 数的入口。

下面将执行本节实现的文件操作案例相关代码，对其中的功能进行演示。

牝鶯演扈动 .

在程序所在文件夹中创建文件Hag,并在其中写入数据“0”，保存退出。执行程序，命令 行将打印如下信息：

首次启动！

用户选择

1. 管理员登录
2. 普通用户登录

请选择用户类型：

此时查看程序所在目录，发现其中新建了文件夹us。花、文件u\_nx)t。在命令行中输入“1”， 进入管理员登录界面，分别输入正确的账户名和密码，程序的执行结果如下所示。

请选择用户类型:1

大\* \* 管理员登";大大大\*\*

请输入账户名:root

请输入密码\* 123456 登录成功！

由以上执行结果可知，管理员的用户名和密码匹配成功。

2再次启动

再次执行程序，命令行将打印如下信息：

欢迎回来！

用户 —

1 ~管理员登录

2-普通用户登录

请选择用户类型：

由以上执行结果可知，c\_fhg()函数调用成功。

本次选择使用普通用户登录，并注册新用户，如下所示。

请选择用户类型，2

是否需要注册? (y/n) : y

用户注册

请输入账户名:itcast

请输入密码？ 123123

请输入昵称:黑马

a普通用*户*登录n

请输入账户*名:*itcast

请输入密码:123123

登录中……

登录成功！

此时打开当前目录下的users目录9可看到其中新建了名为uitcastn的文件。结合以上执

行结果，可知用户注册、普通用户登录都已成功实现。