# 文件操作

# 文件读写

- Python 内置了读写文件的函数,用法和 C 是兼容的。
- 操作系统不允许普通的程序直接操作磁盘,所以,读写文件就是请求操作系统打开一个文件对象(又称文件描述符),然后,通过操作系统提供的接口从这个文件对象操作;

# 文件读写

思考:

把大象放进冰箱的过程。

# 文件读写

思考文件读写的过程:

1. 打开文件

2. 向文件中写入内容;

3. 关闭文件

# 文件读写

```
f = open('/root/hello')

    # 如果文件不存在, open() 函数就会抛出一个 IOError 的错误,并且
```

给出错误码和详细的信息告诉你文件不存在;

# 文件读写

f.read()

#如果文件打开成功,接下来,调用 read() 方法可以一次读取文件的全部内容;

# 文件读写

f.close()

#文件使用完毕后必须关闭,因为文件对象会占用操作系统的资源。

# 文件读写

思考:

read()会一次性读取文件的全部内容,如果文件有 10G,内存就爆了。怎么解决?

# 文件读写

- 如果文件很小, read() 一次性读取最方便;
- 如果不能确定文件大小,反复调用 read(size)
- 比较保险;如果是配置文件,调用 readlines()

# 文件读写

- 二进制文件

  要读取二进制文件,比如图片、视频等等,用 'rb' 模式打开文件即可

```
>>> f = open('/root/test.jpg', 'rb')

>>> f.read()

'\xff\xd8\xff\xe1\x00\x18Exif\x00\x00...' # 十六进制表示的字节
```

# 文件读写

- 字符编码

要读取非 ASCII 编码的文本文件,就必须以二进制模式打开,再解码，Python 还提供了一个 codecs 模块帮我们在读文件时自动转换编码,直接读出 unicode。

```
import codecs

with codecs.open('/Users/michael/gbk.txt', 'r', 'gbk') as f:

        f.read() # u'\u6d4b\u8bd5'
```

# open函数的模式

r 以读的方式打开,定位到文件开头，默认的 mode

r+ 以读写的方式打开,定位文件开头，可以写入内容到文件

w 以写的方式打开,打开文件的时候会清空文件的内容,并且不能读

w+ 以读写的方式打开,定位到文件头,并且打开文件的时候也会清空文件的内容

a 以写的方式打开,定位到文件的末尾,是一个追加的操作，但并不允许读

a+ 以读写的方式打开,定位到文件的末尾,追加的方式。

在使用以上 mode 打开文件的时候,如果增加了b 模式,表示以二进制方式打开

# 文件的其它操作

f.flush()函数，将缓冲区的内容写入到硬盘中

f.seek(offset[,whence])，offset 表示移动多少字节, whence 为 1 的时候表示相对于当前位置移动的;当 2 的时候从文件的末尾往后移动,但不一定所有的平台都支持;默认为 0 表示从文件开头往后移动

f.tell()函数，返回当前文件指针的偏移量:

# 文件的其它操作

fileno() 函数,返回当前的文件描述符,一个数字

isatty() 函数,当前打开的文件是否是一个终端设备

closed 属性,当前文件是否关闭 ,|True,False, f.closed

file 对象是一个迭代器:

next() 方法，一行一行的读，每次读取一行

# with语法

一般情况打开一个文件,经过操作之后,都要显式的执行xx.close() 将文件关闭 .with 用于需要打开、关闭成对的操作,可以自动关闭打开对象 .

with expression as obj:# 将打开的对象赋值给 obj

    expression

    #obj 的作用域只在 with 语句中

over !