[sys.stdin.readline()和raw\_input()的区别](https://www.cnblogs.com/vhh-wei/p/5831052.html)：

sys.stdin.readline()会将标准的输入全部获取，包括末尾的'\n',但是raw\_input()获取的输入是不包括换行符'\n'的。

因此如果在平时使用sys.stdin.readline( )获取输入的话，不要忘了去掉末尾的换行符，可以用strip( )函数去掉（sys.stdin.readline( ).strip('\n')），这样处理一下就行了。

#coding=utf-8 （平安科技）

# 本题为考试多行输入输出规范示例，无需提交，不计分。

import sys

if \_\_name\_\_ == "\_\_main\_\_":

# 读取第一行的n

n = int(sys.stdin.readline().strip())

ans = 0

for i in range(n):

# 读取每一行

line = sys.stdin.readline().strip()

# 把每一行的数字分隔后转化成int列表

values = list(map(int, line.split()))

for v in values:

ans += v

print(ans)

程序代码：

#coding=utf-8  
import sys

# strip()方法只能用于移除字符串 开头和结尾 指定的字符（默认为空格或换行符）或字符序列。

# split(" ")--按空格切分。  
aa=sys.stdin.readline().strip().split(" ") #这种方法好些  
 for i in range(len(aa)-1,-1,-1): # 逆序打印  
#for i in range(len(aa)):  
 print(aa[i],end=' ') #以空格间隔  
# S.join(iterable): 用于把字符串用指定的符号链接起来，返回字符串格式  
# S：需要的分隔符  
# iterable：被分割对象  
# print(" ".join(sys.stdin.readline().strip().split(" ")[::-1])) #可以

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#map(func,list) 有两个参数，前面一个是函数，后面一个是序列。该函数的意义是对一个序列进行前面参数函数的操作，然后返回一个新的list。  
# line= map(int,input("enter character:").split()) # 把一个字符串分割成字符串数组

line= map(int,input().split()) # map类型，要转换为list, 或者用for循环,才能打印出数字，或者字符。  
line= list(line)  
# line= [i for i in line] # 和上面相同效果 ,<class 'list'>

print((line))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# length = len(character)  
# print(length)  
# print(character[length-1])

# print(character[-1]) #和上面效果相同

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
string = sys.stdin.readline().strip().split() # <class 'list'>  
num = 0

方法一：  
for i in string[0]:  
 # print("序号：%s 值：%s" % (string[0].index(i)+1,i))  
 if i==string[1]:  
 num=num+1

elif chr(ord(val)-32)==string[1]: #大小写转换  
 num=num+1  
 elif chr(ord(val)+32)==string[1]: #大小写转换  
 num=num+1

print(num)

方法二：

for i, val in enumerate(string):  
 print("序号：%s 值：%s" % (i + 1, val))  
 if val==string[1]:  
 num=num+1  
print(num)

方法三：

for i in range(len(string[0])):   
 if string[0][i]==(string[1]):   
 num=num+1

chr(x )      将一个整数转换为一个字符

ord(x )      将一个字符转换为它的整数值

chr(65)='A'

ord('A')=65

# upper() 方法将字符串中的小写字母转为大写字母；  
# lower() 方法转换字符串中所有大写字符为小写字母；

# capitalize()将字符串的第一个字母变成大写,其他字母变小写；  
# title() 方法返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sort 与 sorted 区别：

sort 是应用在 list 上的方法，sorted 可以对所有可迭代的对象进行排序操作。

list 的 sort 方法返回的是对已经存在的列表进行操作，无返回值，而内建函数 sorted 方法返回的是一个新的 list，而不是在原来的基础上进行的操作。

sorted(students, reverse=True) # 按降序 False-升序（默认）

while True:  
 try:  
 n=int(input())  
 r=[]  
 for i in range(n):  
 r.append(int(input()))  
 print(r)  
 for i in sorted(set(r),reverse=False): # sorted(r,reverse=True), set()可以去掉重复的数字  
 print(i)  
 except:  
 break

join() 方法：用于将序列中的元素以指定的字符连接生成一个新的字符串。

str = "-";

seq = ("a", "b", "c"); #字符串序列

print str.join( seq );

结果：a-b-c

str.join(sequence)

list=['1','2','3','4','5']

print(''.join(list)) ，‘’表示无空格

结果：12345

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1，在C语言中，设有数组定义：char arrays[]="China"；则数组array所占用的空间为（）。数组array所占的空间为6个字节，最后一个字节存放字符串结束符'\0'。

2，计算机通过MIC(话筒接口)收到的信号是 音频模拟信号 。

MIC(话筒)输出的是音频模拟信号，声卡从MIC获取音频模拟信号后，通过模数转换器(ADC)，将声波振幅信号采样转换成一串数字信号并存储到计算机中。重放时，这些数字信号送到数模转换器(DAC)，以同样的采样速度还原为模拟波形，放大后送到扬声器发声，这一技术称为脉冲编码调制技术(PCM)。

3，栈区：存放局部变量和函数参数值。

寄存器区：存放register变量

代码区：存放函数体中的二进制代码

全局区：存放全局变量和静态变量

堆区：malloc（C）和new（C++）动态申请的内存

1. TCP与UDP区别总结：

TCP/IP由四个层次组成：[网络接口层](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%8E%A5%E5%8F%A3%E5%B1%82" \t "https://blog.csdn.net/qq_34638161/article/details/_blank)、[网络层](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%B1%82" \t "https://blog.csdn.net/qq_34638161/article/details/_blank)、[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "https://blog.csdn.net/qq_34638161/article/details/_blank)、[应用层](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82" \t "https://blog.csdn.net/qq_34638161/article/details/_blank)。

UDP （User Datagram Protocol）， 中文名是**用户数据报协议**，是[OSI](https://baike.baidu.com/item/OSI" \t "https://baike.baidu.com/item/UDP/_blank)（Open System Interconnection，[开放式系统互联](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E5%BC%8F%E7%B3%BB%E7%BB%9F%E4%BA%92%E8%81%94/562749" \t "https://baike.baidu.com/item/UDP/_blank)） 参考模型中一种无连接的[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "https://baike.baidu.com/item/UDP/_blank)协议，提供面向事务的简单不可靠信息传送服务。

与所熟知的TCP（[传输控制协议](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E6%8E%A7%E5%88%B6%E5%8D%8F%E8%AE%AE/9727741" \t "https://baike.baidu.com/item/UDP/_blank)）协议一样，UDP协议直接位于IP（网际协议）协议的顶层。根据OSI（[开放系统互连](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E7%B3%BB%E7%BB%9F%E4%BA%92%E8%BF%9E/10672606" \t "https://baike.baidu.com/item/UDP/_blank)）参考模型，UDP和TCP都属于传输层协议。UDP协议的主要作用是将[网络数据](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%95%B0%E6%8D%AE/509706" \t "https://baike.baidu.com/item/UDP/_blank)流量压缩成数据包的形式。一个典型的数据包就是一个二进制数据的传输单位。每一个数据包的前8个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

（1）、TCP[面向连接](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E8%BF%9E%E6%8E%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接。

（2）、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付

Tcp通过校验和，重传控制，序号标识，滑动窗口、确认应答实现可靠传输。如丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。

（3）、UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

（4）、每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信。

（5）、TCP对系统资源要求较多，UDP对系统资源要求较少。

4.1 HTTP[超文本传输协议](https://baike.baidu.com/item/%E8%B6%85%E6%96%87%E6%9C%AC%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE/8535513" \t "https://baike.baidu.com/item/http/_blank)（HyperText Transfer Protocol)：是[互联网](https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91" \t "https://baike.baidu.com/item/http/_blank)上应用最为广泛的一种[网络协议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/328636" \t "https://baike.baidu.com/item/http/_blank)。HTTP是一个[客户端](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF" \t "https://baike.baidu.com/item/http/_blank)和[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "https://baike.baidu.com/item/http/_blank)端请求和应答的标准（TCP）。客户端是终端用户，服务器端是网站。通过使用[Web浏览器](https://baike.baidu.com/item/Web%E6%B5%8F%E8%A7%88%E5%99%A8" \t "https://baike.baidu.com/item/http/_blank)、[网络爬虫](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB" \t "https://baike.baidu.com/item/http/_blank)或者其它的工具，客户端发起一个到服务器上指定端口（默认[端口](https://baike.baidu.com/item/%E7%AB%AF%E5%8F%A3" \t "https://baike.baidu.com/item/http/_blank)为80）的HTTP请求。（我们称这个客户端）叫用户代理（user agent）。应答的服务器上存储着（一些）资源，比如HTML文件和图像。（我们称）这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个中间层，比如代理，网关，或者隧道（tunnels）。尽管[TCP/IP协议](https://baike.baidu.com/item/TCP/IP%E5%8D%8F%E8%AE%AE" \t "https://baike.baidu.com/item/http/_blank)是互联网上最流行的应用，HTTP协议并没有规定必须使用它和（基于）它支持的层。 事实上，HTTP可以在任何其他互联网协议上，或者在其他网络上实现。HTTP只假定（其下层协议提供）可靠的传输，任何能够提供这种保证的协议都可以被其使用。

ARP：地址解析协议，（Address Resolution Protocol），是根据[IP地址](https://baike.baidu.com/item/IP%E5%9C%B0%E5%9D%80" \t "https://baike.baidu.com/item/ARP/_blank)获取[物理地址](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E5%9C%B0%E5%9D%80/2129" \t "https://baike.baidu.com/item/ARP/_blank)的一个[TCP/IP协议](https://baike.baidu.com/item/TCP/IP%E5%8D%8F%E8%AE%AE" \t "https://baike.baidu.com/item/ARP/_blank)。[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA/455151" \t "https://baike.baidu.com/item/ARP/_blank)发送信息时将包含目标IP地址的ARP请求广播到网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源。地址解析协议是建立在网络中各个主机互相信任的基础上的，网络上的主机可以自主发送ARP应答消息，其他主机收到应答报文时不会检测该报文的真实性就会将其记入本机ARP缓存；由此攻击者就可以向某一主机发送伪ARP应答报文，使其发送的信息无法到达预期的主机或到达错误的主机，这就构成了一个[ARP欺骗](https://baike.baidu.com/item/ARP%E6%AC%BA%E9%AA%97" \t "https://baike.baidu.com/item/ARP/_blank)。[ARP命令](https://baike.baidu.com/item/ARP%E5%91%BD%E4%BB%A4" \t "https://baike.baidu.com/item/ARP/_blank)可用于查询本机ARP缓存中IP地址和[MAC地址](https://baike.baidu.com/item/MAC%E5%9C%B0%E5%9D%80" \t "https://baike.baidu.com/item/ARP/_blank)的对应关系、添加或删除静态对应关系等。相关协议有[RARP](https://baike.baidu.com/item/RARP" \t "https://baike.baidu.com/item/ARP/_blank)、[代理ARP](https://baike.baidu.com/item/%E4%BB%A3%E7%90%86ARP" \t "https://baike.baidu.com/item/ARP/_blank)。[NDP](https://baike.baidu.com/item/NDP" \t "https://baike.baidu.com/item/ARP/_blank)用于在[IPv6](https://baike.baidu.com/item/IPv6" \t "https://baike.baidu.com/item/ARP/_blank)中代替地址解析协议。

ICMP：互联网控制[报文](https://baike.baidu.com/item/%E6%8A%A5%E6%96%87/3164352" \t "https://baike.baidu.com/item/ICMP/_blank)协议（Internet Control Message Protocol），属于网络协议。它是[TCP/IP协议族](https://baike.baidu.com/item/TCP/IP%E5%8D%8F%E8%AE%AE%E6%97%8F" \t "https://baike.baidu.com/item/ICMP/_blank)的一个子协议，用于在IP[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA/455151" \t "https://baike.baidu.com/item/ICMP/_blank)、[路由](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1" \t "https://baike.baidu.com/item/ICMP/_blank)器之间传递控制消息。控制消息是指[网络通](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A" \t "https://baike.baidu.com/item/ICMP/_blank)不通、[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA/455151" \t "https://baike.baidu.com/item/ICMP/_blank)是否可达、[路由](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1/363497" \t "https://baike.baidu.com/item/ICMP/_blank)是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

5， BSS段通常是指用来存放程序中未初始化的[全局变量](https://baike.baidu.com/item/%E5%85%A8%E5%B1%80%E5%8F%98%E9%87%8F/4725296" \t "https://baike.baidu.com/item/BSS%E6%AE%B5/_blank)和[静态变量](https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F/10997955" \t "https://baike.baidu.com/item/BSS%E6%AE%B5/_blank)的一块内存区域。特点是可读写的，在程序执行之前BSS段会自动清0。可执行程序包括BSS段、[数据段](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%AE%B5" \t "https://baike.baidu.com/item/BSS%E6%AE%B5/_blank)、[代码段](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81%E6%AE%B5" \t "https://baike.baidu.com/item/BSS%E6%AE%B5/_blank)（也称文本段）。

一个进程在内存中的布局如图所示：



 其中.text段保存进程所执行的程序二进制文件，.data段保存进程所有的已初始化的全局变量，.bss段保存进程未初始化的全局变量（其他段中还有很多乱七八糟的段，暂且不表）。在进程的整个生命周期中，.data段和.bss段内的数据时跟整个进程同生共死的，也就是在进程结束之后这些数据才会寿终就寝。

栈区：存局部变量，定义的参数，比如int，char之类的，在函数结束时，系统会自动收回存储单元。   
堆区：new,malloc等动态开辟的存在堆，函数结束时，需要我们用delete、free手动释放存储区。

数据节和BSS节都属于数据段。顺便说说对象的存储，可分为三类：静态存储（static storage）；自动存储（automatic storage）；动态分配存储（allocated or dynamic storage）。 对于自动存储则对应的是栈（stack），动态分配存储对应的是堆（heap）； 静态存储可分为.bss/.data/.rodata等数据段（section）。在程序执行中把初始值为零或者是未设初始值的变量放在.[bss段](https://www.baidu.com/s?wd=bss%E6%AE%B5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)中。通过readelf来查看的。

1. 在Linux中普通文件和目录文件保存在称为块[物理设备](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E8%AE%BE%E5%A4%87" \t "https://baike.baidu.com/item/Linux%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F/_blank)的磁盘或者磁带上。一套Linux系统支持若干物理盘，每个物理盘可定义一个或者多个文件系统。（类比于微机[磁盘分区](https://baike.baidu.com/item/%E7%A3%81%E7%9B%98%E5%88%86%E5%8C%BA" \t "https://baike.baidu.com/item/Linux%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F/_blank)）。每个文件系统由逻辑块的序列组成，一个逻辑盘空间一般划分为几个用途各不相同的部分，即引导块、超级块、inode（节点）区以及数据区等。

7、FTP

FTP（File Transfer Protocol，[文件传输协议](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=17202&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)） 是 TCP/IP 协议组中的协议之一。FTP协议包括两个组成部分，其一为[FTP服务器](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=449752&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)，其二为FTP客户端。其中FTP服务器用来存储文件，用户可以使用FTP客户端通过FTP协议访问位于FTP服务器上的资源。在开发网站的时候，通常利用FTP协议把网页或程序传到[Web服务器](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=267249&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)上。此外，由于FTP传输效率非常高，在网络上传输大的文件时，一般也采用该协议。

默认情况下FTP协议使用[TCP端口](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=53267118&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)中的 20和21这两个端口，其中20用于传输数据，21用于传输控制信息。但是，是否使用20作为传输数据的端口与FTP使用的传输模式有关，如果采用主动模式，那么数据传输端口就是20；如果采用被动模式，则具体最终使用哪个端口要服务器端和客户端协商决定。

8、以下那个排序算法的时间复杂度优于O(n^2) ？

插入排序、冒泡排序、简单选择排序、归并排序 ？

其他三个都是O（n^2） 希尔排序，快速排序，堆排序和归并排序是O(nlogn)

1. 列表：用“[]”，元组：用“（）”，字典：用“{}”

10、已知IP地址和子网掩码，求网络地址？

IP（Internet Protocol），意思是“网络之间互连的协议”，也就是为计算机网络相互连接进行通信而设计的协议。在因特网中，它是能使连接到网上的所有计算机网络实现相互通信的一套规则，规定了计算机在因特网上进行通信时应当遵守的规则。IP协议也可以叫做“因特网协议”。常见的IP地址，分为IPv4与IPv6两大类。IP地址被用来给Internet上的电脑一个编号。我们可以把“个人电脑”比作“一台电话”，那么“IP地址”就相当于“电话号码”，而Internet中的路由器，就相当于电信局的“程控式交换机”。

IP地址是一个32位的二进制数，通常被分割为4个“8位二进制数”（也就是4个字节）。IP地址通常用“点分十进制”表示成（a.b.c.d）的形式，其中，a,b,c,d都是0~255之间的十进制整数。例：点分十进IP地址（100.4.5.6），实际上是32位二进制数（01100100.00000100.00000101.00000110）。

子网掩码(subnet mask)又叫网络掩码、地址掩码、子网络遮罩，它是一种用来指明一个IP地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机的位掩码。子网掩码不能单独存在，它必须结合IP地址一起使用。子网掩码只有一个作用，就是将某个IP地址划分成网络地址和主机地址两部分。

（1）首先将网络的IP地址和子网掩码都转换成32位二进制字符。

（2）将转换后的二进制的IP地址和子网掩码进行“逻辑与”运算，得

到新的32位二进制字符。

（3）把得到的新的32位二进制字符转换成十进制，就是主机的网络号了。

本机IP为192.168.1.16 ，子网掩码为255.255.255.0

转换成二进制：

本机IP为11000000.10101000.00000001.00010000

子网掩码11111111.11111111.11111111.00000000

进行“与运算“ 11000000.10101000.00000001.00000000

转换成十进制：192.168.1.0。即为主机的网络号。

11、[进程、线程、多线程相关总结](https://www.cnblogs.com/fuchongjundream/p/3829508.html)：

根本区别：进程是操作系统资源分配的基本单位，而线程是任务调度和执行的基本单位

开销方面：每个进程都有独立的代码和数据空间（程序上下文），程序之间的切换会有较大的开销；线程可以看做轻量级的进程，同一类线程共享代码和数据空间，每个线程都有自己独立的运行栈和程序计数器（PC），线程之间切换的开销小。

所处环境：在操作系统中能同时运行多个进程（程序）；而在同一个进程（程序）中有多个线程同时执行（通过CPU调度，在每个时间片中只有一个线程执行）

内存分配方面：系统在运行的时候会为每个进程分配不同的内存空间；而对线程而言，除了CPU外，系统不会为线程分配内存（线程所使用的资源来自其所属进程的资源），线程组之间只能共享资源。

包含关系：没有线程的进程可以看做是单线程的，如果一个进程内有多个线程，则执行过程不是一条线的，而是多条线（线程）共同完成的；线程是进程的一部分，所以线程也被称为轻权进程或者轻量级进程。

进程的概念主要有两点：

1. 进程是一个实体。每一个进程都有它自己的地址空间，一般情况下，包括文本区域（text region）、数据区域（data region）和堆栈（stack region）。文本区域存储处理器执行的代码；数据区域存储变量和进程执行期间使用的动态分配的内存；堆栈区域存储着活动过程调用的指令和本地变量。
2. 进程是一个“执行中的程序”。程序是一个没有生命的实体，只有处理器赋予程序生命时，它才能成为一个活动的实体，我们称其为进程。

进程状态：进程有三个状态，就绪、运行和阻塞。

就绪状态其实就是获取了出cpu外的所有资源，只要处理器分配资源就可以马上执行。就绪状态有排队序列什么的，排队原则不再赘述。运行态就是获得了处理器分配的资源，程序开始执行。阻塞态，当程序条件不够时候，需要等待条件满足时候才能执行，如等待i/o操作时候，此刻的状态就叫阻塞态。

程序：

程序是指令和数据的有序集合，其本身没有任何运行的含义，是一个静态的概念。而进程则是在处理机上的一次执行过程，它是一个动态的概念。这个不难理解，其实进程是包含程序的，进程的执行离不开程序，进程中的文本区域就是代码区，也就是程序。

线程：

通常在一个进程中可以包含若干个线程，当然一个进程中至少有一个线程，不然没有存在的意义。线程可以利用进程所拥有的资源，在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位，由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统多个程序间并发执行的程度。

多线程：

在一个程序中，这些独立运行的程序片段叫作“线程”（Thread），利用它编程的概念就叫作“多线程处理”。多线程是为了同步完成多项任务，不是为了提高运行效率，而是为了 提高资源使用效率 来提高系统的效率。线程是在同一时间需要完成多项任务的时候实现的。

最简单的比喻多线程就像火车的每一节车厢，而进程则是火车。车厢离开火车是无法跑动的，同理火车也不可能只有一节车厢。多线程的出现就是为了提高效率。

进程与线程的区别：

进程和线程的主要差别：它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

12、