

《程序是怎样跑起来的》（中）



c747190cc2f5 (/u/c747190cc2f5)

2019.05.02 16:48* 字数 1909 阅读 2 评论 0 喜欢 0
(/u/c747190cc2f5)

学习笔记

第4章 熟练使用有棱有角的内存

本章提问

1. 有十个地址信号引脚的内存 IC (集成电路) 可以指定的地址范围是多少?
2. 高级编程语言中的数据类型表示的是什么?
3. 在 32 位内存地址的环境中，指针变量的长度是多少位?
4. 与物理内存有着相同构造的数组的数据类型长度是多少?
5. 用 LIFO 方式进行数据读写的数据结构称为什么?
6. 根据数据的大小链表分叉成两个方向的数据结构称为什么?

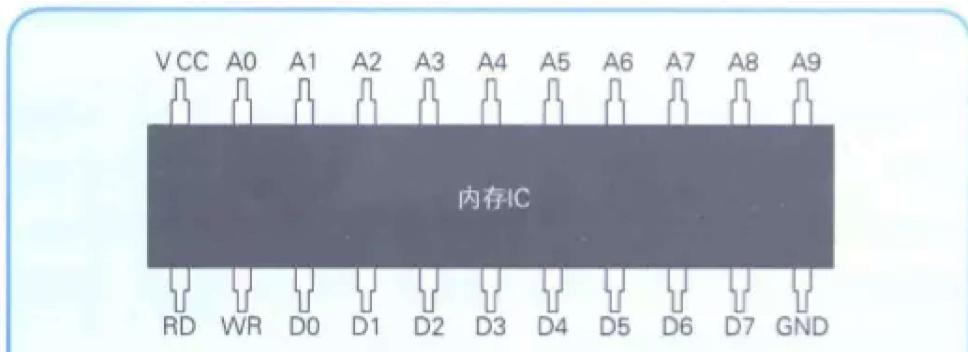
本章重点

对内存要有物理上和逻辑上的认识。



4.1 内存的物理机制很简单

内存实际上是一种名为内存IC的电子元件，包括DRAM、SRAM、ROM多种形式。



内存引脚配置示例

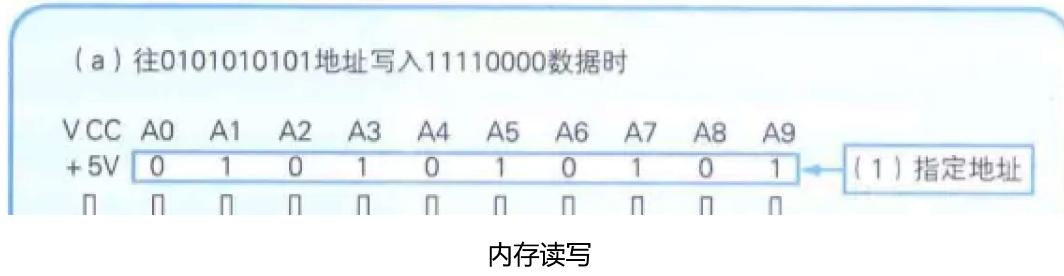
内存可以存储多少数据？

数据引脚有D0~D7共8个，一次可以输入输出8位（1字节）的数据，地址信号有10个，即可表示1024个不同地址，所有内存可以存储1024字节，即1KB的数据。

实际计算机中，会有更多的地址信号引脚，可以存储更多的数据。

内存的读写，改变控制位WR、RD，指定地址即可。





4.2 内存的逻辑模型是楼房

数据类型不同，占用内存大小不同。

地址	内存的内容
0000000000	1字节的数据
0000000001	1字节的数据
0000000010	1字节的数据
⋮	⋮
1111111110	1字节的数据
1111111111	1字节的数据

1024层楼房中，每层都存储着1个字节的数据

1KB内存的模型

地址	内存内容
⋮	⋮
XXXXXXXXXX	123
XXXXXXXXXX + 1	123
XXXXXXXXXX + 2	0
XXXXXXXXXX + 3	123
XXXXXXXXXX + 4	0
XXXXXXXXXX + 5	0
XXXXXXXXXX + 6	0
⋮	⋮

内存地址 × × × × × × × × × × 在程序执行时由OS而定

变量a的范围 = 1字节 (char)
变量b的范围 = 2字节 (short)
变量c的范围 = 4字节 (long)

数据类型不同，占用内存不同



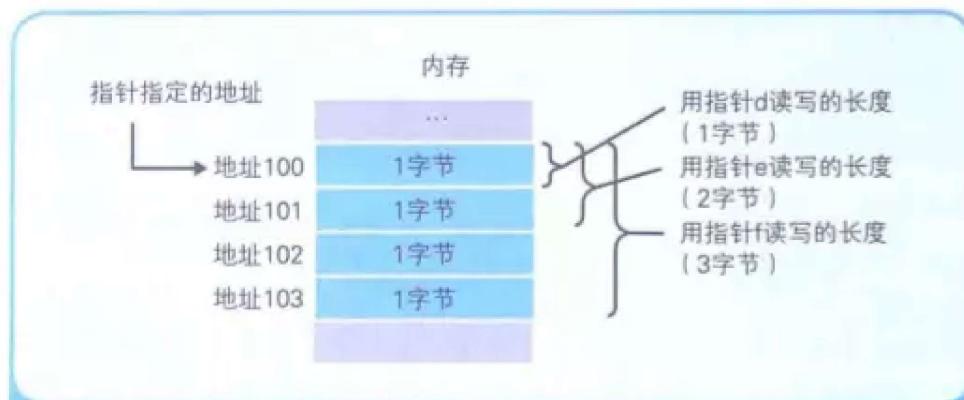
4.3 简单的指针

指针也需要指定数据类型，数据类型表示从指针指向的内存中一次性读取的字节数。

代码清单 4-2 各种数据类型指针的定义

```
char *d;           //char 类型的指针 d 的定义
short *e;          //short 类型的指针 e 的定义
long *f;           //long 类型的指针 f 的定义
```

假设d、e、f都是一百，则使用三个指针从地址为100的内存中读取的字节数不同：



指针的数据类型表示一次可以读写的长度

4.4~4.7

略

问题答案



1. 用二进制数来表示的话是 0000000000 ~ 1111111111 (用十进制数来表示的话是 0 ~ 1023)
2. 占据内存区域的大小和存储在该内存区域的数据类型
3. 32 位
4. 1 字节
5. 栈
6. 二叉查找树 (binary search tree)

解析

1. 用二进制数来表示的话是 0000000000 ~ 1111111111 (用十进制数来表示的话是 0 ~ 1023)

答案

第5章 内存和磁盘的关系

本章问题

问题

5.1 不读入内存就无法运行



程序要加载到内存之后才能运行

5.2 磁盘缓存加快了磁盘访问速度

磁盘缓存指的是把从磁盘中读取的数据存储到内存的一种方式，可以加快数据的访问速度。

但是随着现在磁盘能力的提升，磁盘缓存的作用已经没有那么明显了。

5.3 虚拟内存把磁盘作为部分内存来使用

为了实现虚拟内存，必须把实际内存同虚拟内存进行部分置换，并同时运行程序。

5.4 节约内存的编程方法

两种方法。

第一种方法：通过DLL（Dynamic Link Library，动态链接库）文件实现函数共有

比如有两个应用程序都包含了一个同样的函数：

静态链接

两个应用可以通过DLL文件共有这个函数，从而节约了内存：



动态链接

第二种方法，通过调用_stdcall来减少程序文件大小

关于_stdcall

C语言中，调用完函数需要进行栈清理，比如从main()中调用MyFunc(),按照默认设定，栈清理会附加在main()一方，如果反复调用就会反复清理，造成内存浪费。如果用_stdcall把int MyFunc (int a) 变为int _stdcall MyFunc (int a) ,栈清理就会在MyFunc一方执行，清理一次即可。



5.5 磁盘的物理结构

一般有扇区方式、可变长方式两种，前者长度固定，后者反之。Windows采用扇区方式。

一般一个扇区是512字节，簇是扇区的整倍数，即n扇区。

不同文件不能存储在同一个簇中，不管多小的文件，最少占用一个簇的空间。

如果簇的容量设定过小，那么文件读写就会变慢，如果过大，就会浪费存储空间。所以，簇的大小，是由处理速度和存储容量的平衡来决定的。

问题答案

答案



第6章 亲自尝试压缩数据

本章问题：

问题

6.1 文件以字节为单位保存



文件是字节数据的集合体

6.2 RLE算法的机制

即字节×重复次数

RLE

RLE 算法经常被用于传真 FAX 等。G3 类传真机是把文字和图形都作为黑白图像来发送的。由于黑白图像的数据中，白或黑通常是部分连续的，因此就没有必要再发送这部分数据的值（白或者黑），而只需附带上重复次数即可，这样压缩效率就得到了大幅提升。例如，像白色部分重复 5 次，黑色部分重复 7 次，白色部分重复 4 次，黑色部分重复 6 次这样的部分图像，就可以用 5746 这样的重复次数数字来进行压缩。

关于RLE

6.3 RLE算法的缺点

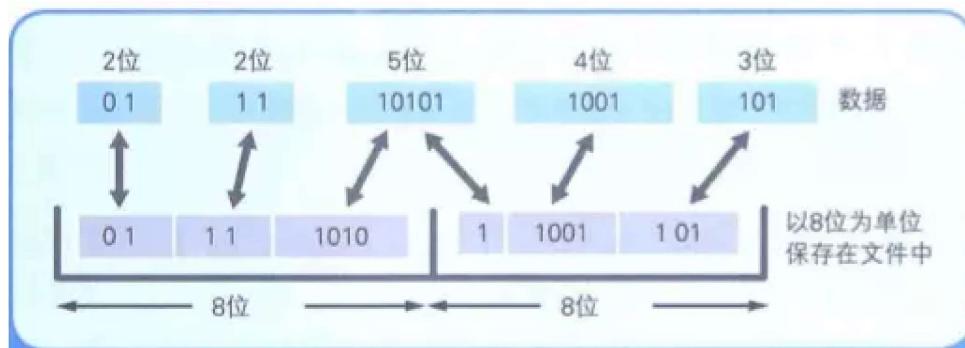
此书作者曾对不同文件做过RLE实验，结果如下：

文件类型	压缩前文件大小	压缩后文件大小	压缩比率
文本文件	14862 字节	29506 字节	199%
图像文件	96062 字节	38328 字节	40%
EXE 文件	24576 字节	15198 字节	62%

可以看出，文本文件的压缩率不降反增，那是因为文本文件中出现重复字符的几率比较小，加入的重复次数反而占据了更多了存储空间。我们可以想办法进行改进，比如不统计单个字符的重复字数，而统计单词的重复次数等等。

6.4 通过摩尔斯编码来看哈夫曼算法的基础

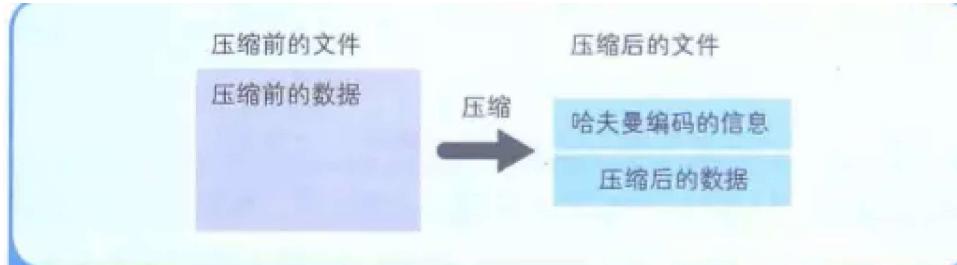
哈夫曼编码的关键就是“常用的字符用较少的比特表示，不常用的字符可以用较多的比特表示”，但要注意不管满不满8个比特，最后都要以8个比特存储在内存中（如下图），这会使程序变得复杂，但是压缩率也会高不少。



摩尔斯电码就不说了。

6.5 用二叉树实现哈夫曼编码

摩尔斯电码的编码体系是固定的，而哈夫曼为不同的压缩对象构建不同的编码体系。



用哈夫曼压缩的文件构造

以AAAAAAABCDDEEEEEEF为例：

字符	出现频率	编码(方案)	位数
A	6	0	1
E	5	1	1
B	2	10	2
D	2	11	2
C	1	100	3
F	1	101	3

编码方案



步骤1：列出数据及其出现频率，()里面表示的是出现频率，
这里按照降序排列

出现频率	(6)	(5)	(2)	(2)	(1)	(1)
数据	A	E	B	D	C	F

步骤2：选择两个出现频率最小的数字，拉出两条线，并在交叉地方
写上这两位数字的和。当有多个选项时，任意选取即可

出现频率	(6)	(5)	(2)	(2)	(1)	(1)
数据	A	E	B	D	C	F

步骤3：重复步骤2，可以连接任何位置的数值

出现频率	(6)	(5)	(2)	(2)	(1)	(1)
数据	A	E	B	D	C	F

步骤4：最后这些数字会被汇集到了1个点上，该点就是根，这样哈夫曼树也就完成了。按照从根部到底部的叶子这一顺序，在左边的树枝（线）处写上0，在右边的树枝（线）处写上1。然后从根部开始沿着树枝到达目标文字后，再按照顺序把通过的树枝上的0或者1写下来，就可以得到哈夫曼编码了

出现频率	(6)	(5)	(2)	(2)	(1)	(1)
数据	A	E	B	D	C	F
哈夫曼编码	00	01	100	101	110	111

构造哈夫曼树

哈夫曼不用在字符之间添加分隔符号，因为字符都是叶子节点，不会重复也不会互相包含（数据结构中有学过）。

6.6 哈夫曼算法能够大幅提高压缩比率



哈夫曼效果

6.7 可逆压缩与非可逆压缩

根据是否能还原为压缩前的状态，分为可逆压缩和非可逆压缩。

程序的exe文件和文本文件的每一个字符都不可丢失，所以要采用可逆压缩；对于图片来说，如果模糊一些也可以接受，则可以采用非可逆压缩。

问题答案



1. 1 字节 (= 8 位)
2. LZH
3. RLE 算法
4. 1 字节 (= 8 位)
5. 没有压缩过
6. 压缩后的数据能复原的是可逆压缩，无法复原的是非可逆压缩

解析 ······

1. 文件是字节数据的集合体。
2. LZH 是用 LHA 等工具压缩过的文件的扩展名。

答案

第7章 程序是在何种环境中运行的

本章问题：

1. 应用的运行环境，指的是什么？
2. Macintosh 用的操作系统 (MacOS)，在 AT 兼容机上能运行吗？
3. Windows 上的应用，在 MacOS 上能运行吗？
4. FreeBSD 提供的 Ports，指的是什么？
5. 在 Macintosh 上可以利用的 Windows 环境模拟器称为什么？
6. Java 虚拟机的功能是什么？

问题



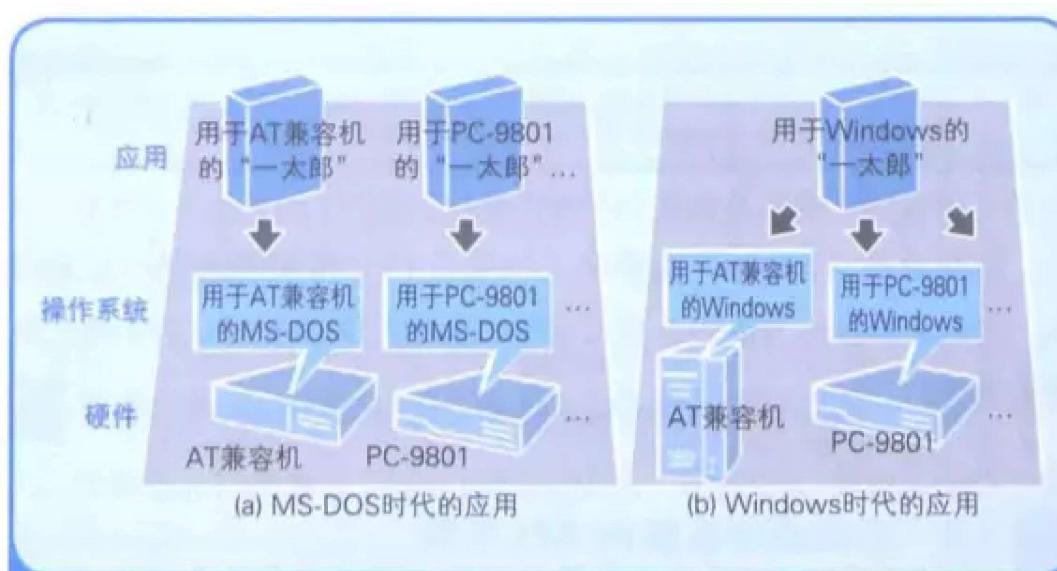
7.1 运行环境=操作系统+硬件

- 操作系统和硬件共同决定运行环境；
- 在硬件中，CPU的种类是特别重要的参数，因为CPU只能解释其自身固有的机器语言，不同的CPU本身的机器语言也是不同的；

- 机器语言的程序称为本地代码，程序员编写的代码称为源代码，市场上出售的一些软件，收录的是本地代码。

7.2 Windows克服了除CPU以外的硬件差异

- Windows本身需要为不同机型提供不同版本，如用于AT兼容机的Windows，用于PC-9801的Windows等；
- Windows克服了除CPU外的硬件差异，这是因为应用程序可以通过Windows间接的向硬件发出指令，而不用直接控制硬件，所以只要Windows正常运行，同样的应用程序（本地代码）在不同的机型上也是可以正常运行的。



Windows在不同机型上使用同一应用

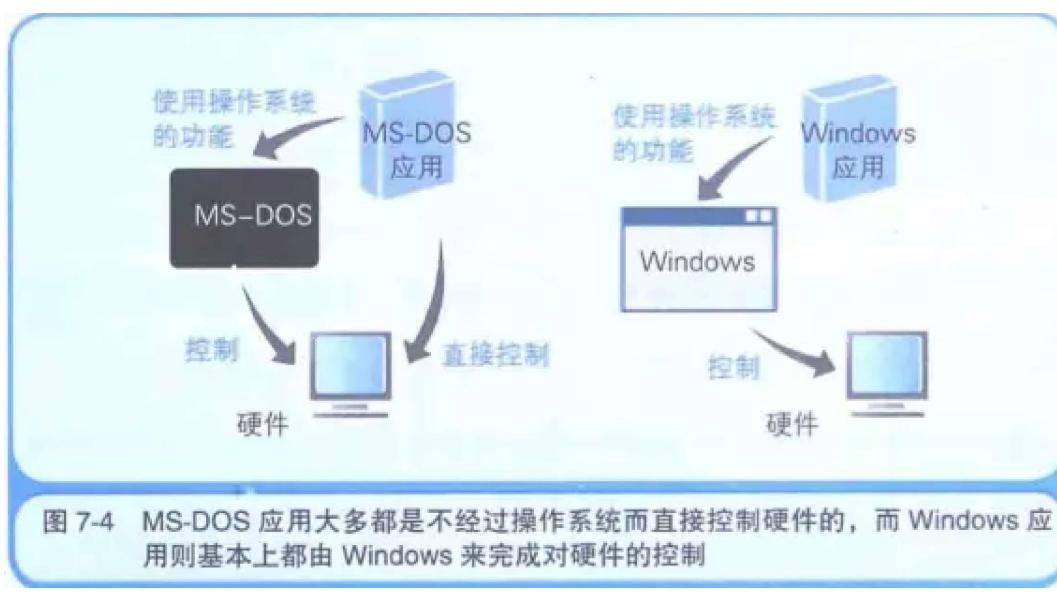


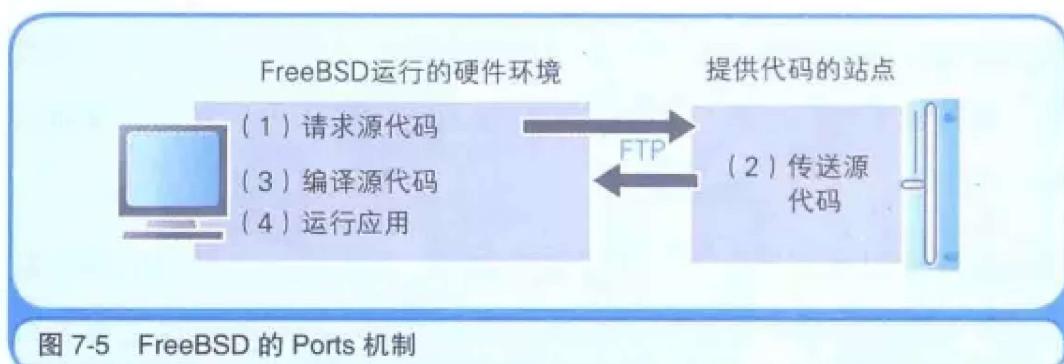
图 7-4 MS-DOS 应用大多都是不经过操作系统而直接控制硬件的，而 Windows 应用则基本上都是由 Windows 来完成对硬件的控制

7.3 不同操作系统的API不同

- 应用程序向操作系统传递指令的途径称为API；
- 不同操作系统的API有差异；
- 将某一操作系统的应用程序移植到另一个操作系统上去，就要重写程序中应用到API的部分。

7.4 FreeBSD Port帮你轻松使用源代码

- FreeBSD是Unix系列操作系统中的一种；
- Ports 表示porting（移植）的意思，是FreeBSD操作系统中的一种机制；
- 该机制会自动下载需要的源代码，然后编译器再生成适合当前环境运行的本地代码；
- 通过Port机制，连CPU在内的硬件差异都可以克服了。

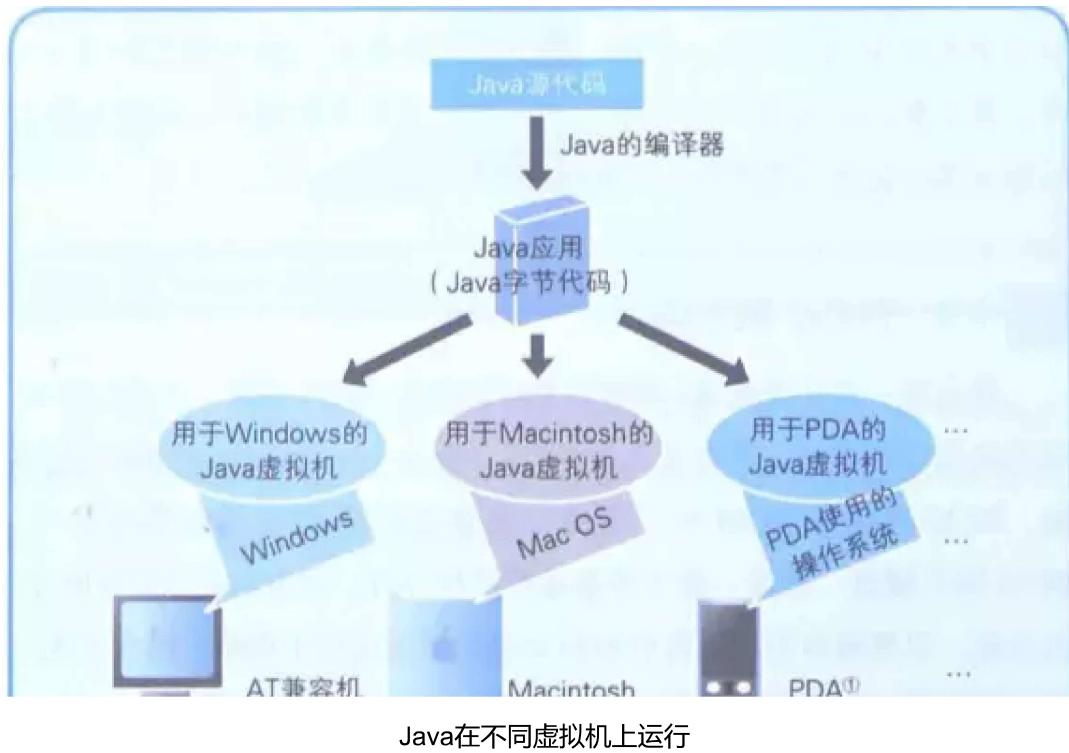


7.5 利用虚拟机获得其它操作系统环境

这个不多讲了

7.6 提供相同运行环境的Java虚拟机

- Java编译后并非生成本地代码，而是生成字节码，字节码运行的环境就是Java虚拟机；
- Java虚拟机可以实现同样的字节码在不同环境下运行；
- 不同Java虚拟机之间无法进行完整互换，想让所有字节码在所有Java虚拟机上任意运行是比较困难的（不太懂）；
- 当我们使用只适用于特定硬件的功能时，就会出现在其它Java虚拟机上无法运行，或者功能受限的情况；
- 解释型语言速度较慢。



7.7 BIOS和引导

- BIOS全称Basic Input/Output System；
- BIOS储存在ROM中，是预先存储在计算机主机上的程序；
- BIOS可以引导操作系统启动

问题答案：



1. 操作系统和计算机本身(硬件)的种类
2. 无法运行
3. 无法运行
4. 通过使用源代码来提供应用，并根据运行环境进行整合编译，从而得以在该环境下运行的机制
5. Virtual PC for Mac

答案

《程序是怎样跑起来的》(下)

(<https://www.jianshu.com/p/4e202e4f35f0>)

小礼物走一走，来简书关注我

赞赏支持

程序是怎样跑起来的 (/nb/36570283)

© 著作权归作者所有



c747190cc2f5 (/u/c747190cc2f5)

写了 25958 字，被 1 人关注，获得了 3 个喜欢

(/u/c747190cc2f5)

一个菜鸡的挣扎...

喜欢



更多分享



写下你的评论...

[评论](#) [关闭评论](#)

智慧如你，不想发表一点想法咩~

