# 阅读本书的感想

# 1.C语言的重要性

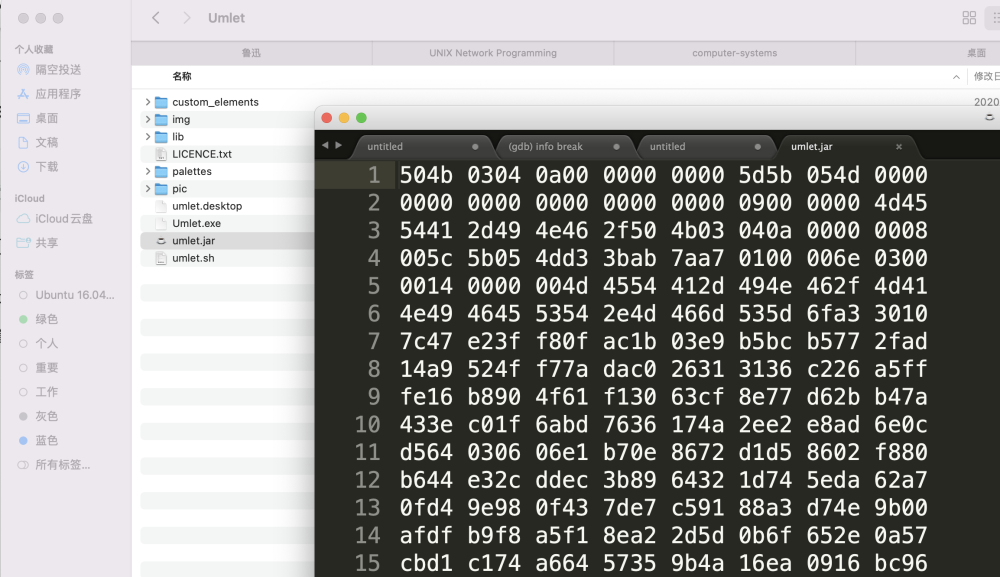
现在知道为啥一上来就要学习C语言了吧？即便在工作中、项目中用到C很少(因为我们一般都是在应用层游荡的CRUD boy)，但是，我们说C/C++还是很重要。这是为啥呢？看了本书就明白了，其中一个原因就是，要介绍本书这种偏底层的内容，用C语言这种天然和底层联系比较亲密的语言，是比较合适的。

随便举一个例子，我们知道，保存在计算机中的数据和程序，都是以10101这种二进制表示的，但是我们打开某个可执行程序，看到的往往是下面这种形式：

备注：这是我们通过记事本打开umlet.jar

这很明显就是16进制的。那么我们在计算机中存储的数据如何转化为16进制呢？书中就用一段C语言程序，解释了如何把int、float、string这些基础数据转化为16进制。

备注：程序具体参考《2.1.3 Addressing and Byte ordering》 show\_bytes.c/show\_bytes.h



这段程序充分体现了C语言的威力。

# 2.浏览目录-体系的思考

光是把本书的三层目录浏览一遍、罗列一遍，就需要花费很长的时间。足见本书整体架构之宏大、体系之完整、细节之深入。什么是体系？这就是体系！这本书涵盖了计算机这个学科大部分的内容。

我们要借此书好好体会。

# 3.看这本书，有一种魔术解密的爽快感。

随便举一个例子，我们知道linux下的管道操作，很神奇对不对：

ls > a.txt

这个是怎么实现的呢？

我们看"10.9 I/O Redirection"就明白原理了：

实现原理就是依托Linux提供的 dup2() function

这个function的实现原理是，在descriptor table中作一下调整：把目标文件的PID也指向源文件，这样目标文件的输出看起来就和源文件的输出保持一致了(本来就已经是同一个文件了)。这个小节中有一幅形象的图来说明这个原理。是不是很神奇？如果脱离这个体系，我无法想象要怎么去清晰、形象地解释管道的原理。

# 4.实践的重要性

实践的重要性，在精读本书的过程中得到了充分体现。

我们例举几个场景。

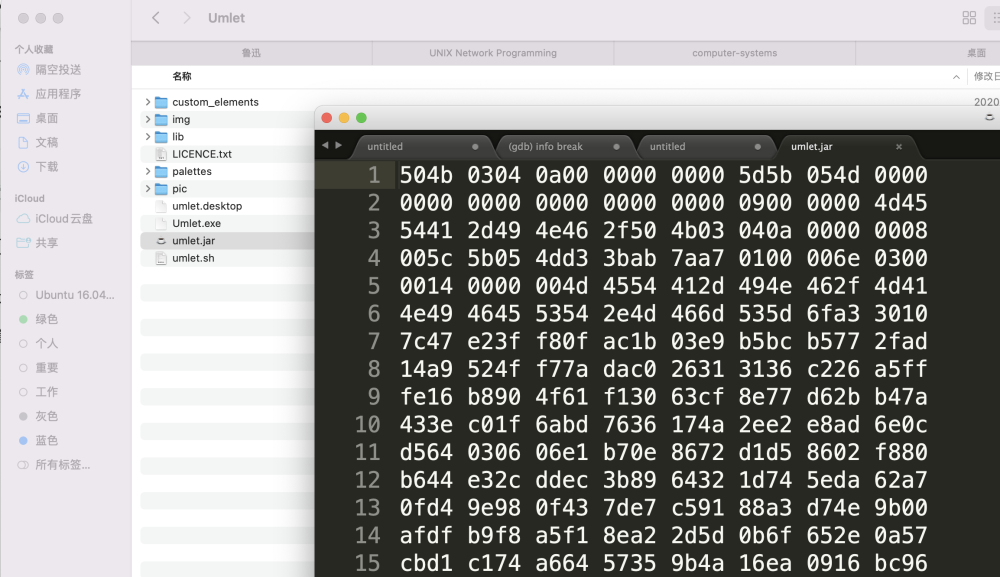
## 场景1 数据、代码的十六进制表示

我们知道，保存在计算机中的数据和程序，都是以10101这种二进制表示的，但是我们打开某个可执行程序，看到的往往是下面这种形式：

备注：这是我们通过记事本打开umlet.jar

这很明显就是16进制的。那么我们在计算机中存储的数据如何转化为16进制呢？书中就用一段C语言程序，解释了如何把int、float、string这些基础数据转化为16进制。

备注：程序具体参考《2.1.3 Addressing and Byte ordering》 show\_bytes.c/show\_bytes.h



我们通过代码实践，搞清楚了数据、代码是如何通过16进制表示的；同时，通过代码逻辑，了解了little-endian和big-endian的区别。

场景1足见代码的重要性。

## 场景2 two's-complementation encodings

《2.2.3 two's-complementation encodings》

引入了负数的表示方式：补码。

小节只介绍了补码的实现方式，但是：

1.正数转化为绝对值相同的负数，二进制形式怎么调整？

2.Two's-complementation方案下的signed number，overflow是怎么回事？

这些都不太清楚。

我们依托代码实践，将signed number转化为二进制，就非常清楚了。

具体代码参考

2.2.3 Two's-complementation encodings/convert-demo.c