中国科学技术大学

计算机导论课程报告

关于计算系统思维的讨论

姓名：王晨沣

学号：PB22061204

班级：信息学院 2202 班

时间：2023 年 6 月 8 日

关于计算系统思维的讨论

王晨沣

1 引言

现今互联网与计算机技术发展进入快车道，越来越多的问题变得复杂而多元化，这使得朴素的线性的处理问题的思维模式受到了很大阻碍。比方一个微信网络消息传递的程序，或是一个很普通的在计算机上运行的软件，如果不能系统化的构建程序的网络、UI、存储等部分的实现过程，或是直接通过指令集来编写一个软件，很显然都是不现实而万分荒唐的。在这种较为大型的项目中，计算系统思维可以很好的解决这种问题。计算系统思维通过将抽象化、模块化、无缝衔接等方法把计算过程分成多个模块化的小问题，将问题的复杂性和多元性大幅降低。

2 定义及核心

计算机科学的研究以及所有的计算过程都需要在计算系统中进行，而设计与理解计算系统的最大挑战便在于计算系统的复杂性。当计算系统较为复杂时，就需要系统的、有条理的、构造计算系统，通过抽象的方法，将模块组合成系统，无缝执行计算过程，此即计算系统思维。

计算系统思维的核心在于将系统抽象成模块，再将模块组合成系统。从计算系统思维的定义和核心可以看出，这种思维可以划分出三个重点：抽象化、模块化、无缝衔接。

其中，抽象化具有三个性质，分别是有限性、精确性、通用性。有限性意味着每个抽象仅需要负责本层的计算内容，而不可以跨层进行设计和优化；精确性意味着每个抽象可以合理地、完全的、规范的表达本层抽象所代表的内容；通用性意味着一个抽象可以解决多个同种的需求，不可复用的抽象是不合适的。

至于模块化，这种方法在设计实践中主要有三个问题需要考虑，分别是模块的划分、模块的连接和模块的执行。模块化的重点在于信息隐藏、接口规范、模块复用。每一个模块分析处理下层信息，并把更为精简的信息传递给上层，其余的信息则对上层进行信息隐藏，这样可以使计算系统设计变得精简化、条理化。每一个模块对其他模块的接口需要有一定的规范，并且这些接口应该和模块内部的实现过程无关，这样可以使得模块的调用、复用更加便捷、易于理解。与抽象化方法类似，模块的可复用性也十分重要，一个不可复用的模块是毫无意义的。

无缝衔接要求计算过程在计算系统中能够流畅的执行，其基于四种原理，其中杨雄周期原理、博斯特尔健壮性原理、冯诺依曼穷举原理被用于分析、处理两个执行过程间的缝隙，而阿姆达尔定律说明了不可改善的瓶颈对系统性能的影响。

3 意义与应用

通过抽象化、模块化、无缝衔接的方法，计算系统思维可以很好地应对系统的复杂性。抽象使计算系统变得条理分明，合理的抽象可以使计算系统应对更多的变化与问题。代码美学也由此而生，一段优美的代码能够有组织的、规范的实现计算过程，具有较好的可读性、可维护性、可拓展性，通过适当的抽象使之没有重复的冗长的代码片段，利用模块化方法使之结构清晰，按照逻辑顺序进行组织，利用无缝衔接的方法处理各个模块中的缝隙，避免代码由于模块间连接不当导致出现异常，并且应妥当处理程序抛出的异常。

几乎所有的计算系统都有涉及或使用计算系统思维相关的方法，下至图灵机、计算机的构成，上至操作系统、微信系统、网络信息的传递，无不涉及抽象与模块。在软件中有数据抽象（比方对数据类型、存储方式的抽象）和控制抽象（比方对算法、程序、进程与指令的抽象），在硬件层面也有控制抽象（比方指令流水线、时序电路、组合电路等）。

4 实例分析

4.1 信息隐藏实验

此处以书中 5.4 节信息隐藏实验为例，分析该系统所涉及的抽象和模块以及无缝衔接的方法。本程序可以用于隐藏信息，能够将一个文本文件的内容隐藏在一张图片里，也可以从隐藏信息的图片中提取隐藏信息。下面给出利用 python 实现的代码。

```
import numpy as np
import PIL.Image as Image

def write_lsb(bin_index, bin_list):
    res = []
    for i in range(8):
        bin_dat = bin(bin_list[i])[2:].zfill(8)
        if bin_index[i] == '0':
            bin_dat = bin_dat[0:7] + '0'
        elif bin_index[i] == '1':
            bin_dat = bin_dat[0:7] + '1'
        res.append(int(bin_dat, 2))
    return res

def read_lsb(bin_list):
    str = ''
    for i in range(len(bin_list)):
        bin_dat = bin(bin_list[i])[2:][-1]
        str += bin_dat
    return str

def hide_proc(img_path,file_to_hide_path):
    img = Image.open(img_path)
    img_data = np.array(img)

    with open(file_to_hide_path) as file:
        txt_to_hide = file.read()
```

```
    imgdat_list = np.array(img.copy()).ravel().tolist()

    img_idx = 0
    res_data = []
    txt_to_hide = "[ENCODED:%d]%s" % (len(txt_to_hide), txt_to_hide)
    for i in range(len(txt_to_hide)):
        index = ord(txt_to_hide[i])
        bin_index = bin(index)[2:].zfill(8)
        res = write_lsb(bin_index, imgdat_list[img_idx * 8: (img_idx + 1) *
8])
        img_idx += 1
        res_data += res
    res_data += imgdat_list[img_idx * 8:]

    new_imgdat_list =
np.array(res_data).astype(np.uint8).reshape((img_data.shape))
    res_im = Image.fromarray(new_imgdat_list)
    res_im.save('res_encode.png')
    print("隐藏成功")

def show_proc(img_path):
    img = Image.open(img_path)
    imgdat_list = np.array(img.copy()).ravel().tolist()

    encoded_head = "[ENCODED:"
    head_len = len(encoded_head)
    head_data = []
    head_str = ""
    for i in range(head_len):
        bin_list = imgdat_list[i * 8: (i + 1) * 8]
        data_int = read_lsb(bin_list)
        head_data.append(int(data_int, 2))

    for i in head_data:
        ch = chr(i)
        head_str += ch
    if head_str != "[ENCODED:":
        print("该图片不含有隐藏信息或隐藏信息不规范")
        return

    secret_start = head_len
    len_data = []
    len_str = ""
    for i in range(head_len, head_len + 9):
        bin_list = imgdat_list[i * 8: (i + 1) * 8]
        data_int = read_lsb(bin_list)
        if int(data_int, 2) == 93: # ']' = 93
            secret_start = i + 1
            break
        len_data.append(int(data_int, 2))
    for i in len_data:
        ch = chr(i)
        len_str += ch
    print(len_str)
    if secret_start == head_len:
        print("该图片不含有隐藏信息或隐藏信息不规范")
```

```
        return

    str_len = int(len_str)
    hidden_data = []
    hidden_str = ''

    for i in range(secret_start, secret_start + str_len):
        bin_list = imgdat_list[i * 8: (i + 1) * 8]
        data_int = read_lsb(bin_list)
        hidden_data.append(int(data_int, 2))

    for i in hidden_data:
        ch = chr(i)
        hidden_str += ch
    print('提取成功，隐藏信息为')
    print(hidden_str)

if __name__ == '__main__':
    while True:
        option = input("1.hide 2.show")
        if option=='1':
            img_path = input('图片背景路径：')
            file_to_hide_path = input('待隐藏文件路径：')
            hide_proc(img_path,file_to_hide_path)
        elif option == '2':
            img_path = input('带有隐藏信息的图片路径：')
            show_proc(img_path)
```

本程序可以实现将一个文件隐藏在图片中，也可以将按特定规范隐藏在图片中的文件显示出来。程序中共定义了四个函数，分别是 write_lsb、read_lsb、hide_proc、show_proc。其中 write_lsb 函数用于将一个字节转化成 8 个比特，并将其嵌入到 8 个像素的最低位中，read_lsb 函数可以从 8 个像素中读取最低位，并整合成一个字节。hide_proc 用于将待隐藏文件中的每个字节都隐藏进图片中，先把图片数据转化为数组，然后将隐藏信息添加信息长度头之后嵌入到图片的 LSB 中。show_proc 用于将图片中的隐藏信息展示出来，先读取头部信息，以获取是否是规范的隐藏图片和隐藏信息长度。

此程序可以很好地体现计算系统思维，其中涉及了各种抽象、模块以及模块之间的无缝衔接。write_lsb 与 read_lsb 是在 hide 与 show 之下的抽象，仅负责对于 lsb 数据的处理，具有有限性；对传入参数和传出数据有着严格的格式规范，具有精确性；在 hide 与 show 函数中多次被调用，并且可以被拓展到其他程序中去，此即泛用性。hide 与 show 整体可以视作一个文件隐藏模块，对外只提供图片路径、文件路径的接口，而将具体的实现过程、文件隐藏规范等等对外进行了信息隐藏，同时这个模块具有实用性和可复用性，可以在其他程序中多次调用，体现了本程序的模块化方法。此外，在 lsb 处理抽象层和文件隐藏抽象层之间也存在着很好的无缝衔接的关系，文件隐藏抽象层通过读取图片文件获取像素序列，并且将像素序列和待隐藏字节传给 lsb 处理抽象层，lsb 层将处理好的像素序列返回给文件隐藏层。

4.2 TCP/IP 衔接关系

TCP 属于传输层协议，基于 IP 协议向上层提供可靠的数据传输，通过面向连接的方式进行可靠的数据包发送与接收。在网络协议层次中，TCP 和 IP 协议分别位于传输层和网络层，其上有应用层、表示层和会话层，其下有数据链路层和物理层。在此仅讨论 TCP 与 IP 之间的接口与衔接关系。

TCP 与 IP 数据报都包括首部和数据部分，TCP 的全部数据报位于 IP 数据报的数据部分中。IP 协议负责地址管理与路由选择，通过 IP 协议发送的数据报可以到达指定的 IP 地址，不过不能保证传输的可靠性，也没有为应用层提供端口服务，如果应用层直连 IP 协议可能导致数据丢失、信息混乱等问题。而通过 TCP 数据报头（图 1）可以看出，TCP 提供了端口服务，并且还有校验和、控制位以确保数据准确。从实现 TCP 协议通信的 socket 库函数可以看出，客户端与服务端在建立连接后仅需要进行 recv 与 send 函数便可以进行可靠的通信，而不需要涉及更为底层的物理层面或以太网层的内容。

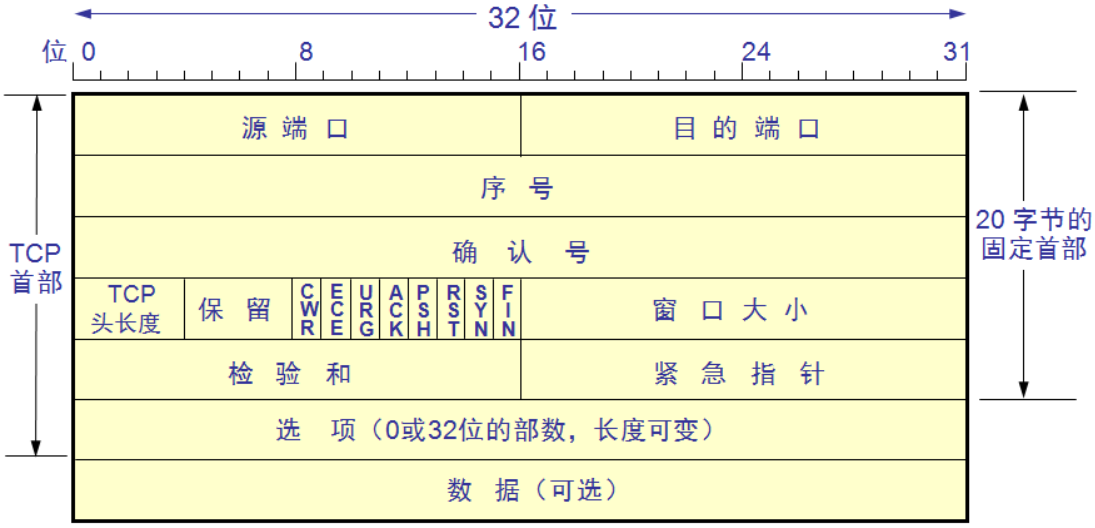

图 1 TCP 数据报头

从上述分析可以看出，TCP 协议是一个很好的抽象层，它具有有限性：仅提供了端口服务、数据传输确认（可靠性），而没有涉及其他层面的内容；精确性：TCP 协议提供了非常规范的数据结构，不会出现数据分析上的歧义；泛用性：在 TCP 之上建立了多种多样的协议，诸如 HTTP、SMTP、TELNET 等协议，这些协议的多样性恰恰证明了 TCP 具有非常强大的泛用性。同时，TCP 协议对外提供了非常简洁的接口，仅需 IP 地址与端口即可进行通信，对外进行了信息隐藏，而对下层则可以进行无缝的衔接。可见 TCP 这一抽象层充分地体现了计算系统思维。

5 结论

本文细致的分析了计算系统思维的定义、特点、意义和应用，结合对一个课本实验和 TCP 协议的研究，笔者对计算系统思维有了更加深刻的理解。在解决问题时，如果能将计算系统思维付诸应用，那将可以收获条理化的思考与规

整的逻辑，大大减少面对复杂计算系统的困难与无措感。

通过本课程的学习以及本次的实验，笔者深深感受到了代码美学的含义。良好的代码有着恰当的抽象、模块与无缝的衔接，使阅读与编写都令人身心愉悦。把构成问题的的各中事务分解为各层抽象，建立抽象的目的不是为了完成某一个事物的某个过程，而是为了描述一类行为或者一类事物。通过对一个个模块的实现、一层层抽象的提取，笔者得到了对计算机编程的各种思维的正确认识，增强了总结能力、学习能力与创新能力，建构了更完整的认知体系与思考模式。

综上，在本次学习中笔者受益良多，笔者会继续学习计算机知识，锻炼计算系统思维，增强学习思维，向成为信息时代的潮流先锋而奋斗。