多看别人的代码，自己写代码时不需要老是钻那种精良的架构，而是强调想出的架构“可完成设计要求\_&\_模块清晰、低耦合”；

至于性能方面”低资源消耗，低耗时“等，多看别人的设计例子，自己多写代码，慢慢摸索；最忌讳的就是，写代码之前一直纠结如何实现最精良的架构，导致迟迟不动手，发现不了问题等等；

“想到设计思路，提出初始方案，完善细节，动手写代码”，如果觉得有必要可以写设计文档，以便往后回顾提升；

时态意识，从过去时开始，当用中文描述过去发生的事，脑子里是提前意识到这是过去发生的（比如昨天），而不是等到要说出口才意识到这是过去发生的，然后再加上“昨天”等字眼；在说中文时如果要描述过去的事，总是很自然加上“昨天”，“刚才”等时间副词，但是英文往往就不可以，was，had这些转换总是没能意识到。

scanf标准输入函数，如果输入的字符串长度多于字符指针分配到的长度（程序变量分配到的长度），那么会报错；C语言输出百分号%不是用转义符，而是printf（"%%"）即可输出一个百分号；C语言数组名编译过去就是首地址指针，但是不能通过数组名加减整数再赋值给数组名（因为数组名代表首地址指针），比如假设数组名arrayA则

arrayA + 1可以（arrayA名仍然代表数组首地址指针，变的是表达式的值）；

arrayA += 1不可以（因为这修改了数组名的“值”），如果非要这么做的话，则新定义一个数组元素类型的指针B，然后B = arrayA；B += 1

直接用read实现（这样只有printf和sprintf用到了标准IO库，或者想办法把那个sprintf也给改了，只用到标准IO库的write就行了）；

还是用输入缓冲区的方式，比如4096个字节，一般情况下是不会超过这个数的，我估计scanf也是这么做的：因为如果标准输入是文件，那么就会有EOF，而如果标准输入是终端（伪终端），好像没有EOF，使用read之后就会一直等待输入，那么它把终端准备好的数据读完也是继续等待输入，因为始终没有收到EOF信息（除非输入CTRL + D）；

如果是标准输入是终端，正常情况下终端一次输入不可能超过数千个字符，如果是文件，那么到文件末尾便是EOF，read会自动停止返回0，所以scanf设置固定大小的缓冲区完全可行；

另外终端最后的enter键即告诉终端准备好数据，同时enter键本身也作为换行符，是准备好的数据的一部分；新的实现方案是“read+4096缓冲区”（应该类似scanf的实现机理）；目前main.c采用sprintf+scanf方面可以限制输入字符，但是遇到空格scanf就要加一些限定符参数之类的，感觉比较麻烦，所以采用上面所说新方案；

n = read(STDIN\_FILENO, input\_buf, BUFFER\_SIZE) == -1；以上是先算==运算符，再算=，所以为了保险，统统加上括号；添加一个使用scanf的注意事项（也许在后续学习Unix环境编程的标准IO库会了解到）

1、scanf("%c", &ch)会将输入打的回车键也处理，即会把回车键放到ch；

2、scanf(" %c", &ch)会忽略后面的回车键；