**基本操作**

C源文件-预处理-编译-汇编-链接-可执行文件

C语言的源文件是以.c为结尾的文件

经过预处理后的文件以.i结尾，所有以#号开头的都是在预处理阶段执行的（gcc -E hello.c > hello.i）

经过编译后以.s结尾，.s文件就是汇编文件（gcc -S hello.i）

再经过汇编之后生成以.o结尾的目标文件（gcc -c hello.s）

最后再经过链接即可生成可执行文件（gcc hello.o -o hello）

一步到位：gcc hello.c -o hello

光标放在函数上，shift+k，立马跳转到函数上去，不需要用man来查看手册了

**头文件正确包含的重要性**

gcc hello.c -Wall：打印出程序中所有的警告

gcc这个编译器比较圆滑，给你的是waring，但可能是error，这样一移植，通篇都是error

要求：gcc给你报的警告，除非你能够解释出来原因，而且你确信这个警告可以忽略，否则，你就要调试到没有警告为止

在C语言中，如果一个函数，我没看到它的原型，那么我就默认它的返回值是int

malloc包含在stdlib.h的头文件中，返回值是void \*（void \*的含义就是你把void \*赋给任何类型的指针都是天经地义的，把任何类型的指针赋给void \*也是天经地义的，有一种特殊情况，后面讲），程序中写了malloc就要包含stdlib.h，否则就会报警告，即使你写了对应的强转操作，警告完全就是你没包含头文件，找不到函数原型，如果我没见到原型，那么这个函数的返回值一般情况下会被当做int来对待，你加强转完全没有必要，也没有道理

重复要求，把gcc报出来的内容调到没有警告为止，除非剩下的是你能解释出来的警告，碰到错误不要慌，错误往往就出现在警告中，把所有警告打印出来，解决了警告就解决了错误

新手用return 0，老手用exit(0)，exit包含在stdlib.h头文件中

**以函数为单位来进行程序编写**

C语言以main作为入口，以main作为出口

能独立出来的功能就把它独立成一个小函数，模块化编程

**声明部分+实现部分**

养成C的好习惯

变量先定义后使用，函数先声明后实现

**return 0；**

return 0;本身含义是结束当前函数，exit(0);的含义是结束当前进程，这两放在main函数中没有区别，返回值给父进程看的，谁创建了它，谁就是它的父进程，比如前面的hello.c，它的父进程就是shell，return 0给谁看的呢，一定要理解为给父进程看的，别说shell，因为未必你这个程序就是在shell环境下运行的

在c语言数据结构中，习惯性写法，如果程序执行成功返回为0，不成功的话一般带回一个非0值来描述各种各样的出错情况，这不是默认，而是程序员的一种约定俗成的做法而已，它的真假一般和shell脚本是对立的

printf的返回值，就是它要打印的字符个数

echo $?，如果返回0表示上一条命令执行成功，非0就表示执行失败

**多用空格空行！（重要）**

**添加注释**

注释是给自己和队友看的，不注释，三天就忘

//:单句

/\*

\*函数定义开头

\*多行

\*注释

\*/

你觉得某个函数有问题，不想让它参与编译

#if 0

func()

#endif

友情提示：写程序时的注释，最好全用英文，这样可以不涉及到中文的标识和英文标识了（标点符号）

**算法**

打开冰箱门，把大象装进去，关上冰箱门

画流程图或NS图或有限状态机将算法描述出来

有限状态机FSM（好东西，通信，应用层，找工作有优势）

F：finite，有限的

S：state，状态

M：machine，机器

**尽最大可能防止写越界，防止内存泄漏，谁打开谁关闭，谁申请谁释放，能做到这四点，你就是个很合格的程序员了**

**C语言其实没什么难的，其实抽象出来，没有任何定义或者是值的分配能够跳过下面这个公式的**

**TYPE NAME = VALUE;**