## 第一章

1. C标准 ([C参考手册](https://zh.cppreference.com/w/c))

第一版公认的C标准：《C语言程序设计》

C89/C90——第一个ANSI/ISO C标准

名称：ISO/IEC 9899:1990

1983年组建委员会(X3J11)，开发一套新标准，于1989年正式公布。

该标准定义了C语言和C标准库，国际标准化组织于1990年采用了这套C标准(ISO C)

ISO C和ANSI C是完全相同的标准。

ANSI/ISO标准的最终版本通常叫C89(因为ANSI于1989年批准该标准)

又叫C90(因为ISO于1990年批准该标准)

又由于ANSI先公布C标准，因此业界人士通常使用ANSI C

C95

这是对 C89 的一个修订和扩充，称为“C89 with Amendment 1”或 C95，严格说来并不是一个真正的标准。

C99

名称：ISO/IEC 9899:1999

1994年ANSI/ISO联合委员会(C9X委员会)开始修订C标准，最终发布了C99标准

C11

名称：ISO/IEC 9899:2011

标准委员会在2007年承诺C标准的下一个版本C1X，2011年发布C11标准。

C99的一些特性成为C11的可选项

C17

名称：ISO/IEC 9899:2018

又称 C17/C18 （分别代表完成和出版的年份

C23

未发布

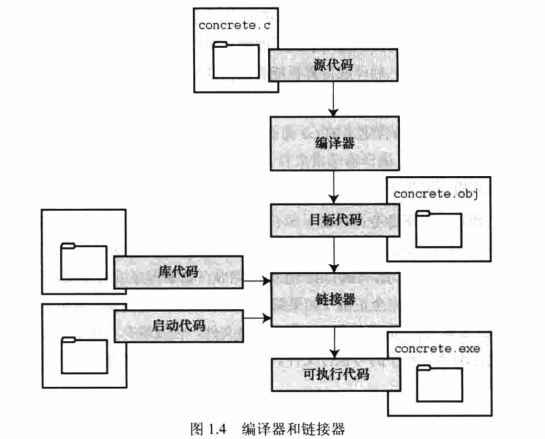
注：

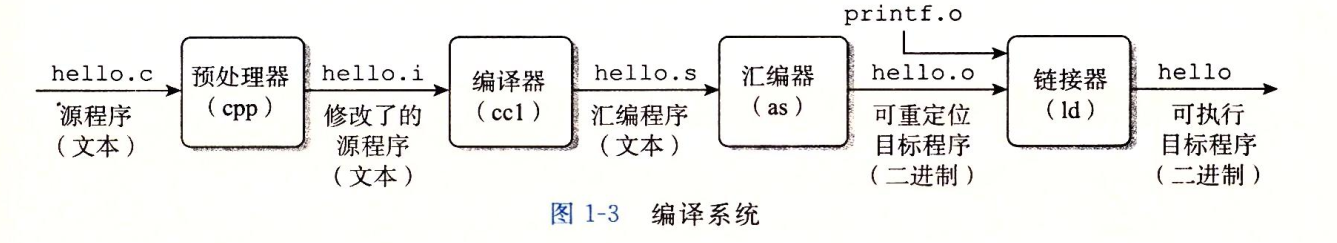
ISO：International Organization for Standardization 国际标准化组织

IEC：International Electro technical Commission 国际电工委员会

ANSI：American National Standards Institute 美国国家协会

2. 程序编译过程





预处理阶段：预处理器(cpp)根据字符#开头的命令，修改原始的C程序。比如#include <stdio.h>命令告诉预处理器读取系统头文件stdio.h的内容，并把它直接插入程序文本中。结果就得到了另一个C程序，通常是以.i作为文件扩展名

编译阶段：编译器(ccl)将文本hello.i翻译成文本文件hello.s，它包含一个汇编语言程序。该程序包含函数mian的定义。

汇编阶段：汇编器(as)将hello.s翻译成机器语言指令，把这些指令打包成一种叫做可重定位目标程序的格式，并将结果保存在目标文件hello.o中。hello.o文件时一个二进制文件。在文本编辑器中直接打开将显示一堆乱码。

链接阶段：hello程序调用了库函数printf，printf函数存在于一个名为printf.o的单独预编译好了的目标文件中，而这个文件必须以某种方式合并到我们的hello.o程序中。链接器(ld)就负责这种合并。最后就得到hello文件，这个文件是一个可执行文件，可被加载到内存中，由系统执行。

这里我用GCC来举例：[GCC编译工具详解](https://zhuanlan.zhihu.com/p/351841622" \l ":~:text=GCC%E7%BC%96%E8%AF%91%E5%B7%A5%E5%85%B7%E9%93%BE%E6%98%AFGCC%E7%BC%96%E8%AF%91%E5%99%A8%E7%9A%84%E4%B8%80%E6%95%B4%E5%A5%97%E5%B7%A5%E5%85%B7%EF%BC%8C%E4%B8%BB%E8%A6%81%E4%BD%9C%E7%94%A8%E5%B0%B1%E6%98%AF%20%E5%B0%86%E6%BA%90%E4%BB%A3%E7%A0%81%E8%BD%AC%E5%8C%96%E4%B8%BA%E5%8F%AF%E6%89%A7%E8%A1%8C%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F%EF%BC%81%20%E6%97%A9%E6%9C%9F%20GCC%20%E7%9A%84%E5%85%A8%E6%8B%BC%E4%B8%BA%20GNU%20C%20Compiler%EF%BC%8C%E5%8D%B3,GUN%20%E8%AE%A1%E5%88%92%E8%AF%9E%E7%94%9F%E7%9A%84%20C%20%E8%AF%AD%E8%A8%80%E7%BC%96%E8%AF%91%E5%99%A8%EF%BC%8C%E6%98%BE%E7%84%B6%E6%9C%80%E5%88%9D%20GCC%20%E7%9A%84%E5%AE%9A%E4%BD%8D%E7%A1%AE%E5%AE%9E%E5%8F%AA%E7%94%A8%E4%BA%8E%E7%BC%96%E8%AF%91%20C%20%E8%AF%AD%E8%A8%80%E3%80%82)

链接阶段补充：动态库与静态库

3.Linux、GNU、GNU/Linux

Linux与GNU/Linux

Linux严格来说是单指操作系统的内核，因操作系统中包含了许多用户图形接口和其他实用工具。如今Linux常用来指基于Linux的完整操作系统，内核则改以Linux内核称之。由于这些支持用户空间的系统工具和库主要由理查德·斯托曼于1983年发起的GNU计划提供，自由软件基金会提议将其组合系统命名为GNU/Linux，但Linux不属于GNU计划，这个名称并没有得到社群的一致认同。

GNU/Linux

GNU是一个自由的操作系统，其内容软件完全以GPL方式发布。这个操作系统是GNU计划的主要目标，名称来自GNU's Not Unix!的递归缩写，因为GNU的设计类似Unix，但它不包含具著作权的Unix代码。GNU的创始人，理查德·马修·斯托曼，将GNU视为“达成社会目的技术方法”。作为操作系统，GNU的发展仍未完成，其中最大的问题是具有完备功能的内核尚未被开发成功。GNU的内核，称为Hurd，是自由软件基金会发展的重点，但是其发展尚未成熟。在实际使用上，多半使用Linux内核、FreeBSD等替代方案，作为系统核心，其中主要的操作系统是Linux的发行版。Linux操作系统包涵了Linux内核与其他自由软件项目中的GNU组件和软件，可以被称为GNU/Linux（但GNU/Linux命名存在争议）。  
 GNU计划，又称革奴计划，是由Richard Stallman在1983年9月27日公开发起的。它的目标是创建一套完全自由的操作系统。Richard Stallman最早是在net.unix-wizards新闻组上公布该消息，并附带一份《GNU宣言》等解释为何发起该计划的文章，其中一个理由就是要“重现当年软件界合作互助的团结精神”。

GNU是“GNU's Not Unix”的递归缩写。Stallman宣布GNU应当发音为Guh-NOO，与canoe发音相同，以避免与gnu（非洲牛羚，发音与new相同）这个单词混淆。UNIX是一种广泛使用的商业操作系统的名称。由于GNU将要实现UNIX系统的接口标准，因此GNU计划可以分别开发不同的操作系统部件。GNU计划采用了部分当时已经可自由使用的软件，例如TeX排版系统和X Window视窗系统等。不过GNU计划也开发了大批其他的自由软件。

为保证GNU软件可以自由地“使用、复制、修改和发布”，所有GNU软件都在一份在禁止其他人添加任何限制的情况下授权所有权利给任何人的协议条款，GNU通用公共许可证（GNU General Public License，GPL）。这个就是被称为“反版权”（或称Copyleft）的概念。

1985年Richard Stallman又创立了自由软件基金会（Free Software Foundation）来为GNU计划提供技术、法律以及财政支持。尽管GNU计划大部分时候是由个人自愿无偿贡献，但FSF有时还是会聘请程序员帮助编写。当GNU计划开始逐渐获得成功时，一些商业公司开始介入开发和技术支持。当中最著名的就是之后被Red Hat兼并的Cygnus Solutions。

到了1990年，GNU计划已经开发出的软件包括了一个功能强大的文字编辑器Emacs，C语言编译器GCC，以及大部分UNIX系统的程序库和工具。唯一依然没有完成的重要组件就是操作系统的内核(称为HURD)。

1991年Linus Torvalds编写出了与UNIX兼容的Linux操作系统内核并在GPL条款下发布。Linux之后在网上广泛流传，许多程序员参与了开发与修改。1992年Linux与其他GNU软件结合，完全自由的操作系统正式诞生。（尽管如此GNU计划自己的内核Hurd依然在开发中，目前已经发布Beta版本。）

许多UNIX系统上也安装了GNU软件，因为GNU软件的质量比之前UNIX的软件还要好。GNU工具还被广泛地移植到Windows和Mac OS上。

总结：Linux单指linux内核，GNU是一个计划其目标是创建一套完全自由的操作系统但是其内核没完全做好，所以一般用的是linux的内核，而其他的系统工具则是GNU计划提供的。并且有很多系统的发行版，所以其他人就称这样的系统为GNU/Linux但是并没有得到广泛的认可此名称存在争议。  
  
参考资料：[GNU、Linux和GNU/Linux之间的关系](https://www.cnblogs.com/nerrissa/articles/5613739.html)

4.编译器

GNU项目始于1987年，是一个开发大量免费UNIX软件的集合。GNU编译器集合(也被称为GCC，其中包含GCC C编译器)是该项目的产品之一。其编译器紧跟C标准改动。GCC有各种版本以适应不同的硬件平台和操作系统，包括Linux、UNIX、和Windows。

命令行编译器：

MinGW(Minimalist GNUfor Windows)。它是一个可自由使用和自由发布的Windows特定头文件和使用GNU工具集导入库的集合，允许你在GNU/Linux和Windows平台生成本地的Windows程序而不需要第三方C运行时（C Runtime）库。MinGW 是一组包含文件和端口库，其功能是允许控制台模式的程序使用微软的标准C运行时（C Runtime）库（MSVCRT.DLL）,该库在所有的 NT OS 上有效，在所有的 Windows 95发行版以上的 Windows OS 有效，使用基本运行时，你可以使用 GCC 写控制台模式的符合美国标准化组织（ANSI）程序，可以使用微软提供的 C 运行时（C Runtime）扩展，与基本运行时相结合，就可以有充分的权利既使用 CRT（C Runtime）又使用 WindowsAPI功能。

集成开发环境（IDE，Integrated Development Environment ）:

是用于提供程序开发环境的应用程序，一般包括代码编辑器、 编译器 、 调试器 和 图形用户界面 等工具。

## 第二章

C99和C11允许使用更长的标识符名，但编译器只识别前63个字符。

外部标识符只允许使用31个字符

标识符命名规则：

大小写字母、数字、下划线来命名

名称第一个字符必须是字符或下划线，不能是数字

C语言中的的6种语句

* 标号
* 复合
* 表达式
* 选择
* 迭代
* 跳转 比如return

提高程序的可读性

可读性高的程序是良好的编程习惯。可读性高的程序更容易理解，以后也更容易修改和更正。提高程序的可读性还有助于你理清编程思路。

提高程序可读性的技巧：

1.选择有意义的函数名

2.写注释

3.在函数中用空行分隔概念上的多个部分

4.每条语句各占一行

注：函数及变量命名法则

驼峰命名法：指混合使用大小写字母来构成变量和函数的名字，其实我们工作比较常用的就是驼峰命令法

例：printEmployeePaychecks();

匈牙利命名法：广泛应用于象Microsoft Windows这样的环境中。Windows 编程中用到的变量(还包括宏)的命名规则为匈牙利命名法，这种命名技术是由一位能干的 Microsoft 程序员查尔斯-西蒙尼(Charles Simonyi) 提出的。

匈牙利命名法通过在变量名前面加上相应的小写字母的符号标识作为前缀，标识出变量的作用域、类型等。这些符号可以多个同时使用，顺序是先m\_(成员变量)、再指针、再简单数据类型、再其它。这样做的好处在于能增加程序的可读性，便于对程序的理解和维护。

例如：m\_lpszStr, 表示指向一个以0字符结尾的字符串的长指针成员变量。

匈牙利命名法关键是：标识符的名字以一个或者多个小写字母开头作为前缀；前缀之后的是首字母大写的一个单词或多个单词组合，该单词要指明变量的用途。

帕斯卡命名法：与驼峰命名法类似，二者的区别在于：驼峰命名法是首字母小写，而帕斯卡命名法是首字母大写

例：DisplayInfo();

string UserName；

下划线命名法：

例：print\_employee\_paychecks();

补充：

用正确的反义词组命名具有互斥意义的变量或函数等

add/remove

begin/end

create/destroy

insert/delete

first/last

get/release

increment/decrement

put/get

add/delete

lock/unlock

open/close

min/max

old/new

start/stop

next/previous

source/target

show/hide

send/receive

source/destination

cut/paste

up/down

C关键字



## 第三章