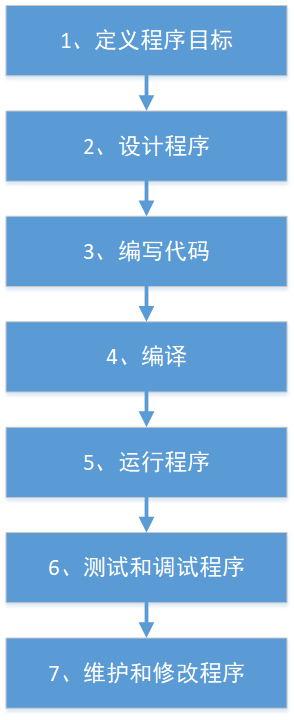
1、基本概念

C程序是包含声明的一系列文本文件，通常是包含头文件和源文件，源文件的拓展名为.c，它们能经过翻译变成可执行程序，在操作系统调用其主函数时被执行，除非它自己就是OS程序或者其它独立程序，这时候入口是由编译器所决定。

2、编程基本步骤

C是一种编译性的语言，编写程序的过程主要分解为7个步骤，如下所示：



简要概述这7个步骤，如下：

（1）定义程序目标

在开始编程之前需要对程序程序做什么有一个清晰的想法，考虑程序所需要的信息、计算和操作、程序需要报告的信息，规划时使用一般概念考虑问题。

（2）设计程序

在对程序有一定的概念认识后，接下来决定程序如何完成它，用户界面应该是什么样的，程序应该如何组织，目标用户是谁，有多长时间去完成程序，确定程序如何表示数据，用什么方法来处理数据。

（3）编写代码

对程序有了清晰的设计后，开始通过编写代码去实现它，将前面的程序设计解释为C语言。

（4）编译

接下来就是编译源代码，通过编译器将源代码转换为可执行代码，编译过程中会检查程序是否是有效的C语言程序，如果编译过程中发生错误，将错误报告。

（5）运行程序

在一定的环境下运行编译成功的可执行程序。

（6）测试和调试程序

程序运行后，不一定运行得正确，这时候应该进行检查，看程序是否在做要做的事情，调试就是要发现并修正程序的错误。

（7）维护和修改程序

当为自己或别人创建程序时，该程序可能会有更广泛的应用，如果是这样的话，可能会发现需要对其进行更改，这时候就需要维护和修改编写的程序。

编程实际上并不会像上面的一条线过程，往往是在不同的步骤间来回反复，应该养成在编写代码前先进行规划的习惯，使用笔记略记下程序的目标，并勾勒出设计概貌。

3、程序编译

假定使用C语言编写了一个简单的程序，文件名为app.c，源文件代码如下所示：

//app.c代码

对于这个app.c源文件，是怎么样经过编译和链接成可以在系统中运行的可执行程序呢？

（1）目标代码文件、可执行文件和库

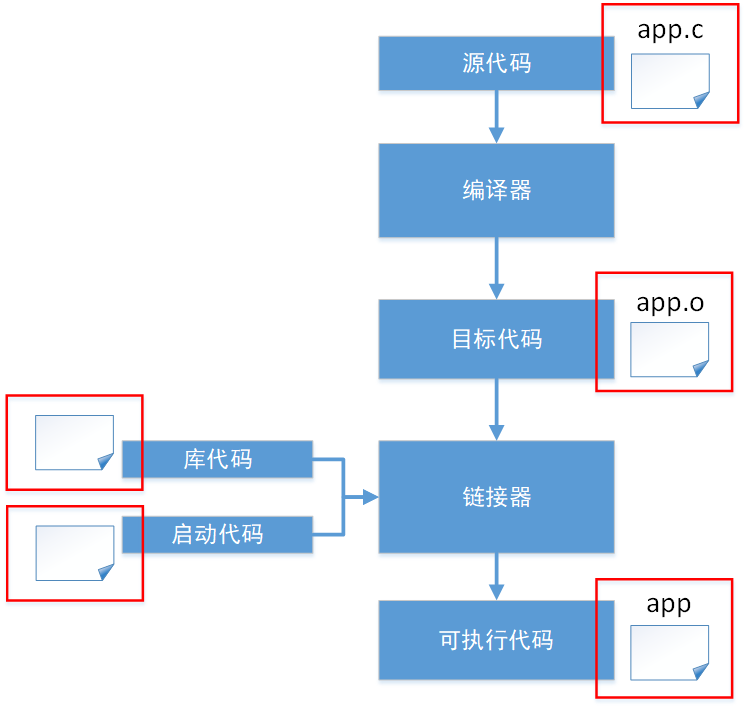
C编程的基本策略是使用程序将源代码文件转换为可执行文件，也就是机器可以运行的机器语言代码，这个工作分为两个步骤：编译和链接，其中编译再细分的话，可以分为预处理、编译和汇编，编译器将源代码转换为中间代码，连接器将中间代码与其它代码相结合生成可执行文件，同过这样划分，可以便于程序模块化，在多个模块编程下，其中一个模块改变并不需要重新编译所有其它模块。

在上面也提到了，编译细分下来还有3步，因此在编译的过程中，中间文件的形式是多种的，如果将源代码文件转换为机器语言代码，将结果放到一个目标代码文件，其实就是目标文件（.o或者.obj），此时该目标文件并不能直接运行，和最终能在机器上运行的程序相比较，还缺少相关的启动代码和库代码。

启动代码相当于所编写的程序和操作系统之间的接口，通过该部分代码能在系统中去执行你所编写的程序，另外，我们一般编写的C程序都会利用到标准C库中的函数，例如前面编写的app.c源文件，就使用到了标准C库函数printf()，目标文件代码中并不包含该函数的代码，实际的代码存储在另一个”库”文件中。

链接这个过程实际上就是将目标代码、系统的标准启动代码以及库代码这3个元素进行结合在一起，并生成一个可执行文件。

对于app.c这个源文件经过编译器和链接器生成可执行文件app的整个过程如下：



app.o就是app.c源代码编译出来的目标代码文件，app就是最终的可执行程序。

（2）在Linux系统上编译

如果我们在Linux系统上编写了c源码程序的话，可以在系统终端上使用gcc工具进行程序编译链接，gcc是一个非常强大的工具，全称为GNU Compiler Collection，支持多种编程语言编译，例如上面编写的app.c源文件，可以直接使用下面的命令生成可执行程序，可执行程序为默认名称a.out：

$ gcc app.c

当然，你也可以使用-o选项指定最终生成可执行程序的名称，如定义为app，使用下面命令：

$ gcc -o app app.c

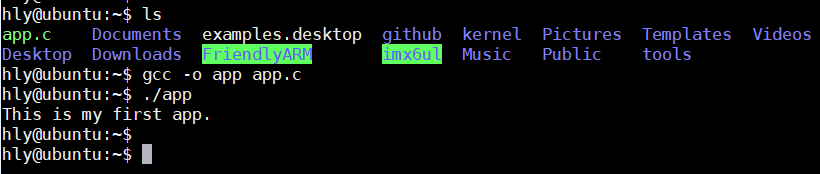
在Linux系统上运行编程出来的可执行程序，使用下面命令：

$ ./a.out

or

$ ./app

命令执行效果如下所示：



关于gcc编译的详细过程，可以参考下面文章，链接如下：

<https://www.cnblogs.com/Cqlismy/p/11496053.html>

4、小结

本文主要简单介绍了C的一些基本概念，包含C的一般编程步骤和C源程序是如何通过编译链接成系统上可以运行的可执行程序。