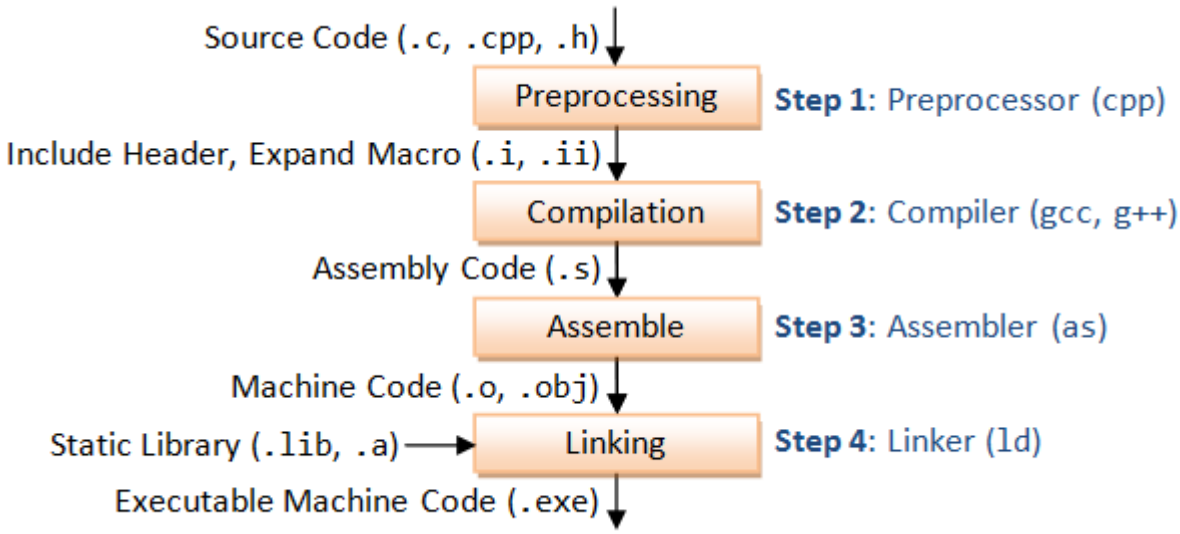
计算概论（C语言）习题课讲义03

内容概要

- 从源代码到可执行程序
- 变量的使用
- 函数的使用

从源代码到可执行程序

更为具体流程如下图所示，



1. 预处理(Preprocessing) 使用预处理器`cpp` (也可使用`gcc -E`)，将所有的`#include`头文件以及宏定义替换成其真正的内容。预处理之后得到的仍然是文本文件，但文件体积会大很多。
2. 编译(Compilation) 使用编译器`gcc` (或者`gcc -S`)，将经过预处理之后的文件转换成特定的汇编代码。
3. 汇编(Assemble) 使用汇编器`as` (也可以使用`gcc -c`)，将上一步的汇编代码转换成机器码(machine code)，这一步产生的文件叫做目标文件，是二进制格式。
4. 链接(Linking) 使用链接器`ld`，将多个目标文以及所需的库文件(.so等)链接成最终的可执行文件(executable file)。

演示：从源代码到程序

> 名词解释：GCC
>
> the GNU Compiler Collection Link
>
> The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Ada, Go, and D, as well as libraries for these languages (libstdc++,...). GCC was originally written as the compiler for the GNU operating system.

变量的使用

一个变量是一个固定大小的“仓库”，用于存储一定类型的数据。变量的定义是“申请仓库”的过程，变量的赋值或取值，是“使用仓库”的过程。值得注意的是，C语言对局部变量，并不“主动打扫仓库”。

演示: 变量的未初始化

> 名词解释：局部变量和全局变量
>
> 不严格地说，函数内部的变量被称为局部变量(Local Variable)，它的作用域仅限于函数内部，离开该函数后就是无效的，再使用就会报错。而在所有函数外部定义的变量称为全局变量(Global Variable)，它的作用域默认是整个程序

演示：局部变量和变量的区别

函数的使用

一个函数是一组共同完成某个任务的语句集合。

函数的定义

```
return_type function_name( parameter list )
{
   body of the function
}
```

函数的声明

演示：函数不声明的错误

```
return_type function_name( parameter list );
```

例如，在使用printf等库函数前必须包含对应头文件的原因，其实就是头文件中包含了对应函数的函数声明。

演示：**stdio.h**文件查看

函数的调用

当程序调用函数时，程序控制权会转移给被调用的函数。当函数的返回语句被执行时，或到达函数的结束括号时，会把程序控制权交还给主程序。

演示：函数调用的流程控制

函数的参数传递

如果函数要使用参数，则必须声明接受参数值的变量。这些变量称为函数的形式参数。 形式参数就像函数内的其他局部变量，在进入函数时被创建，退出函数时被销毁。

形参的传递方式	描述
值传递	该方法把参数的实际值复制给函数的形式参数。在这种情况下，修改函数内的形式参数不会影响实际参数。
指针传递	通过指针传递方式，形参为指向实参地址的指针，当对形参的指向操作时，就相当于对实参本身进行的操作。

默认情况下，C使用*值传递*来传递参数，这意味着函数内的代码不能改变用于调用函数的实际参数。

演示：函数值传递验证

main函数

main函数地位特殊，它表示这个程序的执行起点和整个过程；关于main函数的定义，不同的书上可能有如下两个版本：

```
int main()
```

```
int main(int argc, char *argv[])
```

第二种写法实际上指明了main函数均具有argc和argv这两个形式参数。

演示：**main**函数形参的使用

参考链接