跟涛哥一起学嵌入式 20：一段C语言代码编译、运行全过程解析

文档说明	作者	日期
来自微信公众号：宅学部落(armLinuxfun)	wit	2019.10.21
嵌入式视频教程淘宝店：https://wanglitao.taobao.com/		
联系微信：brotau(宅学部落)		

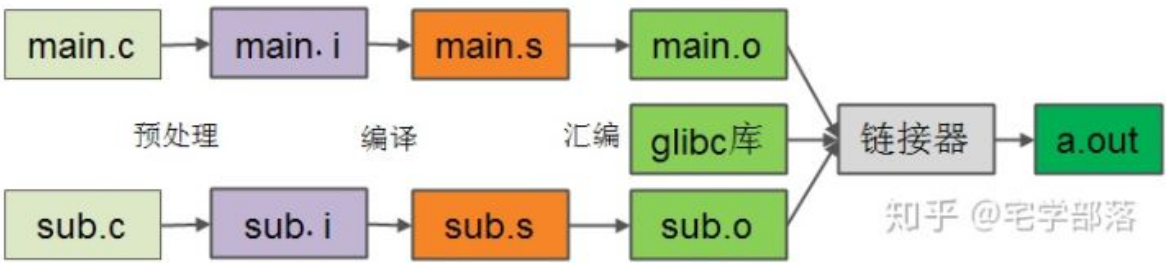
很多嵌入式初学者，不明白一个简单的C语言程序，是如何通过一步步编译、运行变成一个可运行的可执行文件的，程序到底是如何运行的？运行的过程中需要什么环境支持？

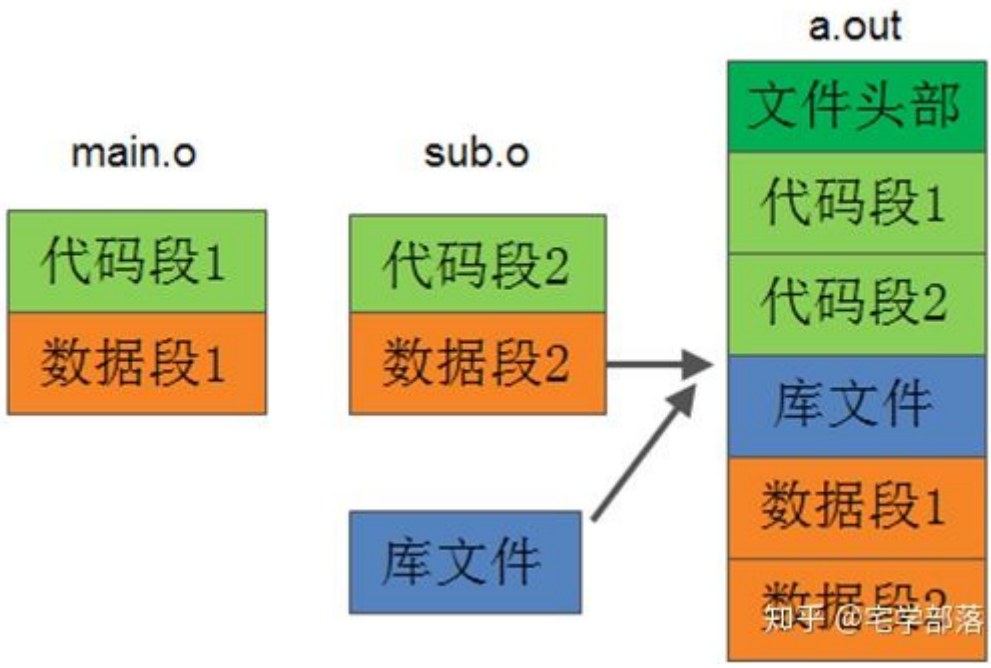
今天就跟大家一起捋一捋这个流程，搞清程序编译、链接、加载、运行的整个脉络，以及在运行过程中的内存布局、堆栈变化。

1. 程序的编译、链接过程

就以hello.c为例：从一个C语言源文件，到生成最后的可执行文件，基本流程如下;

1. C 源文件： 编写一个简单的helloworld程序
2. 预处理：生成预处理后的C源文件 hello.i
3. 编译：将C源文件翻译成汇编文件 hello.s
4. 汇编：将汇编文件汇编成目标文件 hello.o
5. 链接：将目标文件链接成可执行文件





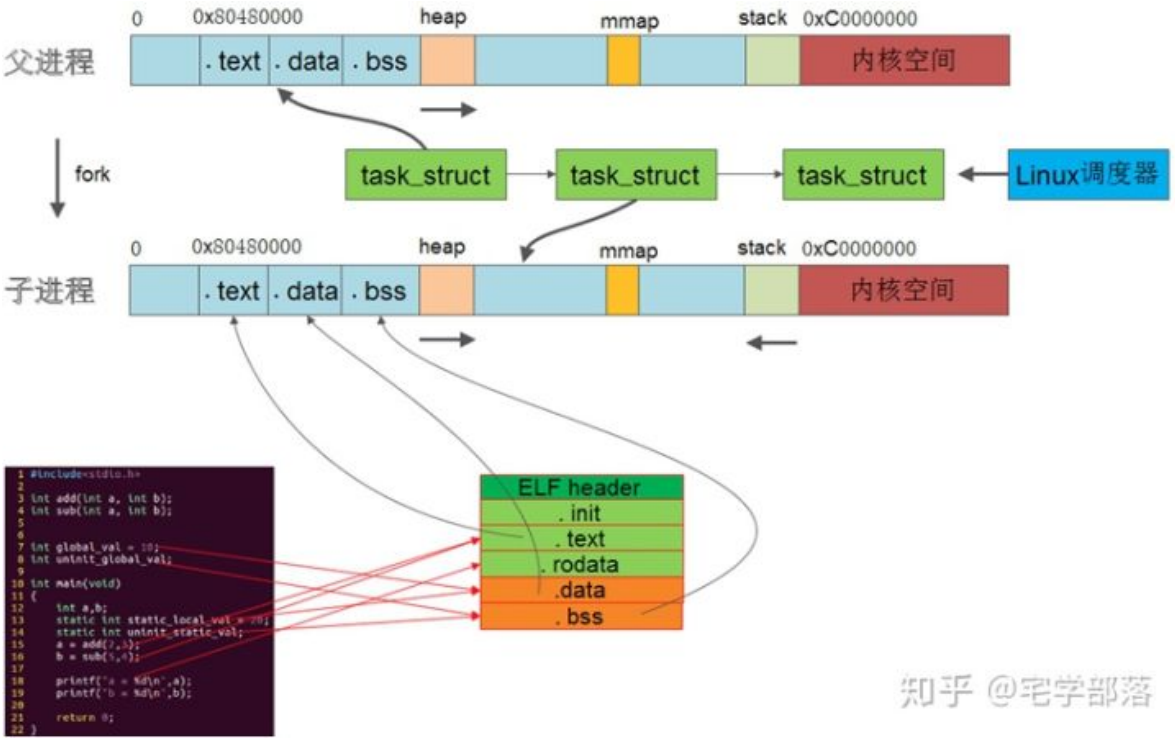
为了加深对这个过程的理解，我们可以在Linux环境下面，通过gcc命令精确控制每一个编译、链接过程

```
$  gcc  -E  hello.c  >  hello.i          //会生成预处理后的C源文件hello.i
$  gcc  -S  hello.i                      //将hello.i编译成汇编文件hello.s
$  gcc  -c  hello.s                      //将汇编文件hello.s汇编成hello.o
$  gcc hello.o  -o hello                 //将目标文件链接成可执行文件hello
$  ./hello                              // 运行可执行文件hello
```

2. 程序的执行过程

当我们在shell交互环境下敲击 $ ./hello，这个hello程序到底是怎么运行的呢？

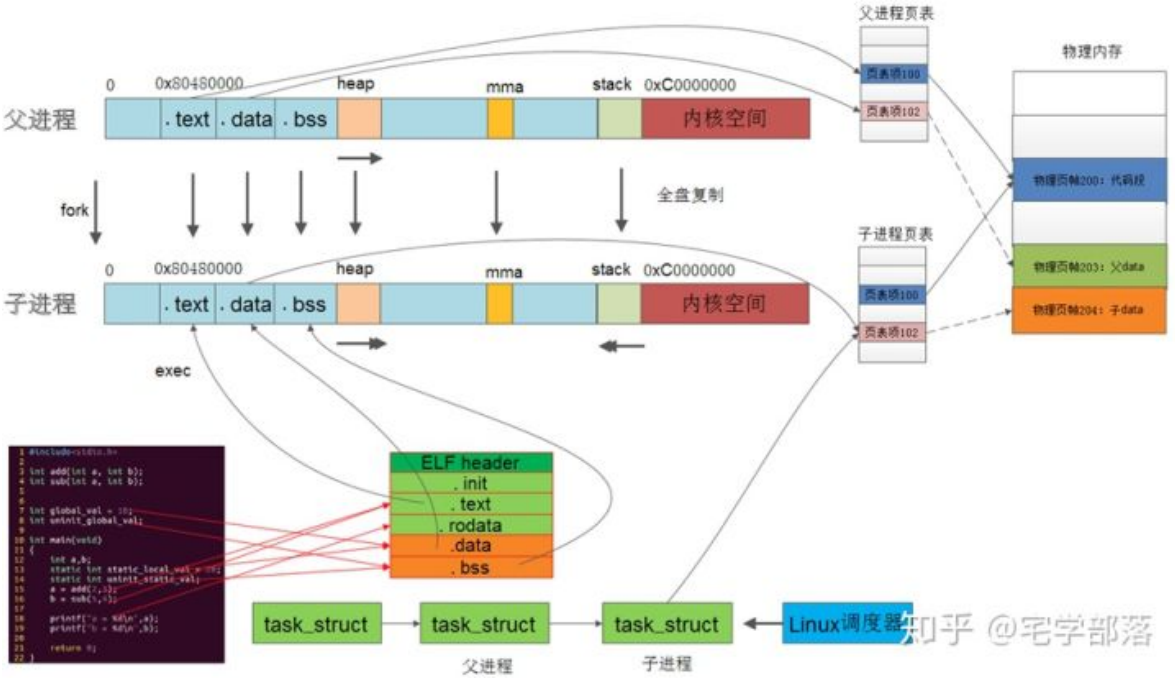
很简单。shell会首先通过系统调用fork创建一个子进程，然后从磁盘上将可执行文件hello的代码段、数据段加载(map)到这个子进程的地址空间内，接下来，在操作系统调度器的调度下，各个进程轮流占用CPU，就可以直接执行了。



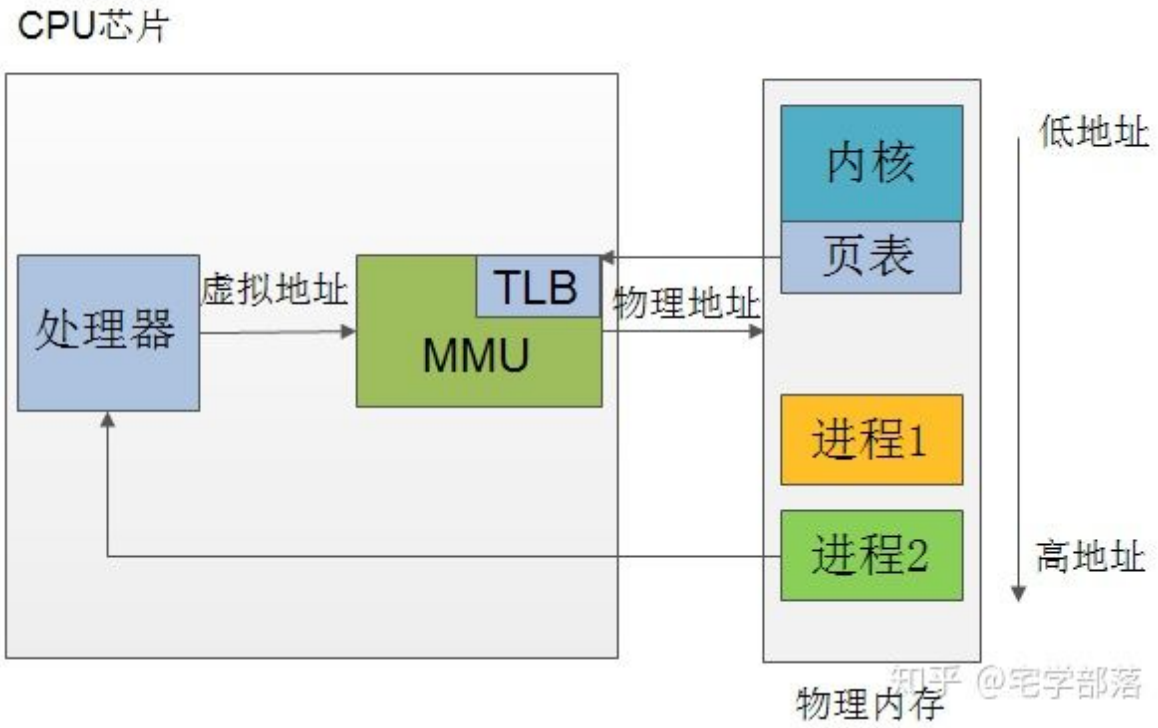
在操作系统层面，对于每一个进程，在内核中都会有一个task_struct的结构体来描述它，里面存储进程的各种信息，各个结构体构成一个链表，操作系统通过调度器来轮流执行每个进程，如上图所示。

3. 进程的虚拟空间和物理空间

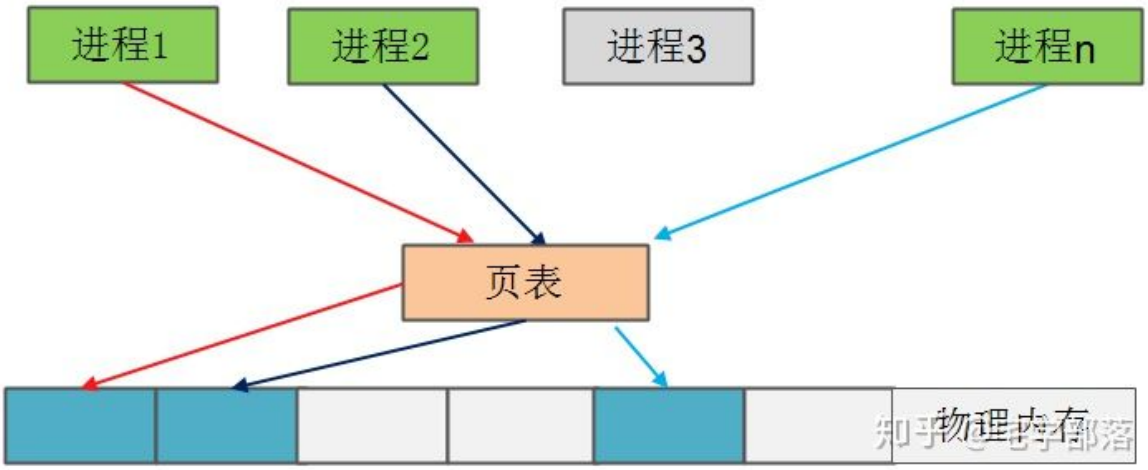
每个进程使用的都是虚拟地址，地址空间0~4G，都是相同的。但是CPU在实际执行过程中，对于每个进程相同的虚拟地址，会映射到物理内存中的不同位置。每个进程都有自己的进程页表，在这个页表里有该进程虚拟地址和物理地址的对应关系。



CPU内部有一个叫MMU的硬件部件会根据这个映射关系，直接将虚拟地址转换成物理地址，如下图所示。



使用虚拟地址的好处之一就是：为每个进程提供一个独立的、私有的物理地址空间，保护每个进程的空间不会被其它进程破坏。同时通过MMU对内存读写权限进行管理、保障系统的安全运行。如下图所示，每个进程在我们的物理内存(DDR)上，都有各自独立的内存空间：一个进程崩溃了，一般情况下，不会影响到系统，不会影响到其它进程的运行。



4. 进程栈

栈是C语言运行的基础。没有栈，C语言函数是无法运行的：这是因为函数调用过程中的返回地址、参数传递、函数内的局部变量都是在栈中存储的，没有栈，C语言函数就无法运行。

Linux进程中的代码也是由一个个函数组成的，所以在运行进程之前，我们要首先初始化栈，如下图所示：

在程序运行过程中，通过栈指针，我们就可以将函数内的局部变量、返回地址保存在栈中。随着函数不断地调用、函数退出，而不断地入栈、出栈。

栈是一种数据结构，CPU的寄存器一般来讲，在设计的时候，会自动入栈出栈、自动增减栈的地址。比如ARM中的入栈出栈操作，当我们使用push/pop入栈出栈的时候，CPU的寄存器SP，即栈指针会自动增减地址，一直指向栈顶，这些都是指令集的实现，即CPU内部硬件电路的实现。关于栈的进一步解释，可以看看我以前的回答：https://www.zhihu.com/question/346428264/answer/836122985

5. 用户栈、内核栈、中断栈

在Linux环境下，进程一般分为两种模式，用户态和内核态。甭管是什么态，只要你是C语言，运行C代码就必须指定栈，否则C代码就无法运行。所以栈又分为用户栈和内核栈。

用户栈的虚拟地址空间在用户空间，内核栈的地址在内核空间。它们都是虚拟地址，最后通过MMU映射到物理内存的不同区域。

有时候，你还会看到中断栈的字眼，千万别被它吓到。中断程序、中断函数也是C语言，也是妖他妈生的，想运行中断处理程序也必须需要栈的支持，一般这种栈叫做中断栈。它可以使一个独立的中断栈，也可以占用进程栈的空间，跟进程栈共享。

6. 小结

以上只是简单介绍一下一个C语言从编译、链接、运行、到进程创建、内存堆栈的大致流程。实际过程比这个更复杂一些、更深一些，限于篇幅的关系，很多细节无法一一细讲。

以上文字和图片，是根据《C语言嵌入式Linux高级编程》视频教程改编而成。想进一步深入学习，推荐一套视频教程：从计算机架构、CPU指令集、编译原理、堆栈内存管理、Linux多进程调度等角度，全方位阐述一个程序的编译、链接、运行的整个过程：https://item.taobao.com/item.htm?spm=a2oq0.12575281.0.0.25911debq7bpfr&ft=t&id=577829845886

专注嵌入式、Linux精品教程：https://wanglitao.taobao.com/
嵌入式技术教程博客：http://zhaixue.cc/
联系 QQ：3284757626
嵌入式技术交流QQ群：475504428
微信公众号：宅学部落(armlinuxfun)