跟涛哥一起学嵌入式 24：Linux进程间通信10分钟快速入门

文档说明	作者	日期
来自微信公众号：宅学部落(armLinuxfun)	wit	2020.3.8
嵌入式视频教程淘宝店：https://wanglitao.taobao.com/		
联系微信：brotau(宅学部落)		

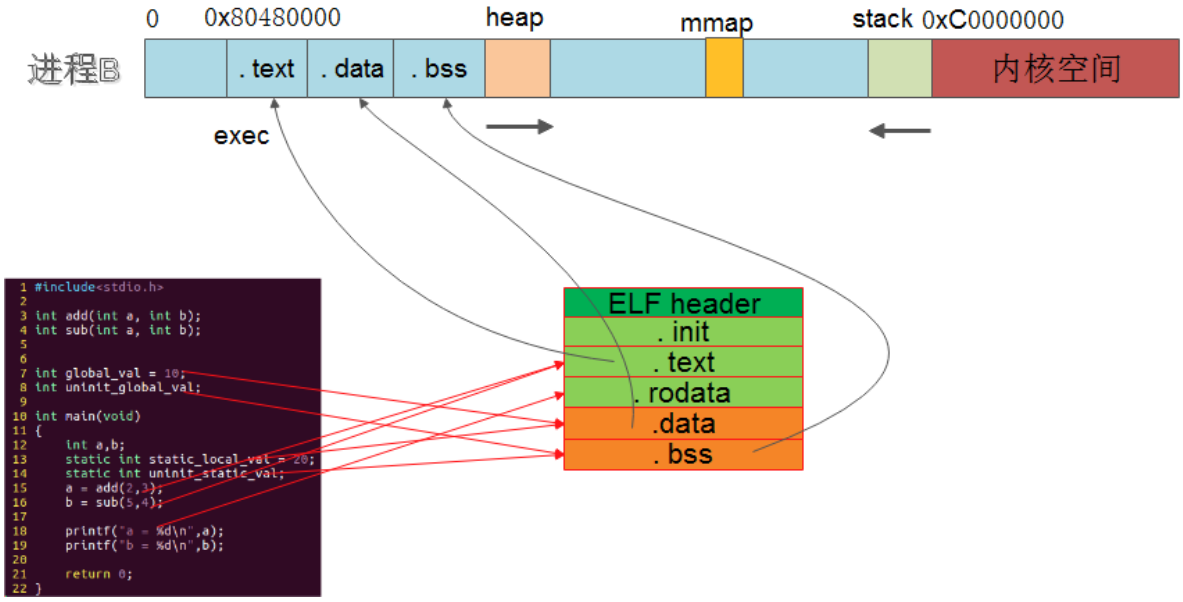
在Linux运行下程序，无论是点击桌面上的一个图标，还是在命令行下敲击一个shell命令，Linux系统都会把我们的程序“包装”成一个进程的形式，然后调度运行：每个进程轮流占用CPU一段时间去执行，时间到了就让给其它进程，时间片轮转，只要轮转得速度足够快，就会给用户一种错觉：我们在电脑上一边听歌，一边打字，感觉像多个程序在同时运行。不同进程在运行过程中，根据自己的需要，进程相互之间也会通信：比如传输数据、发送信号等。

在Linux环境下的进程间通信(inter-process communication，简称IPC)有多种工具可以使用，比如：无名管道pipe、命名管道FIFO、消息队列、共享内存、信号量、信号、文件锁、socket等。这些IPC工具以系统调用或库函数API的形式提供给用户使用：可以在不同的进程之间传输数据、同步、或者发送信号。比如，我们可以使用ctrl+C组合键去终止一个进程，或者使用shell命令kill 3567去杀死一个进程pid为3567的进程，其实都是给进程发送信号，进程接收信号并响应信号的过程。

不同的IPC工具，使用场合不同，也有各自的优劣。为了更好地使用它们，我们不仅要熟练掌握API接口的使用，还要对它们的通信机制、实现原理有一个大致的了解。只有掌握了底层的实现机制、才能明白每个IPC通信工具的优点和缺点、以及他们的使用场合。想要真正理解Linux进程之间到底是如何通信的，首先要搞明白Linux的不同进程在运行过程中，在内存中是以什么样的形态存在的，以及与Linux内核之间是如何交互的。想要理解这点，我们还需要对Linux下程序的编译、执行过程有一个大概的了解。

1. 程序的编译和执行

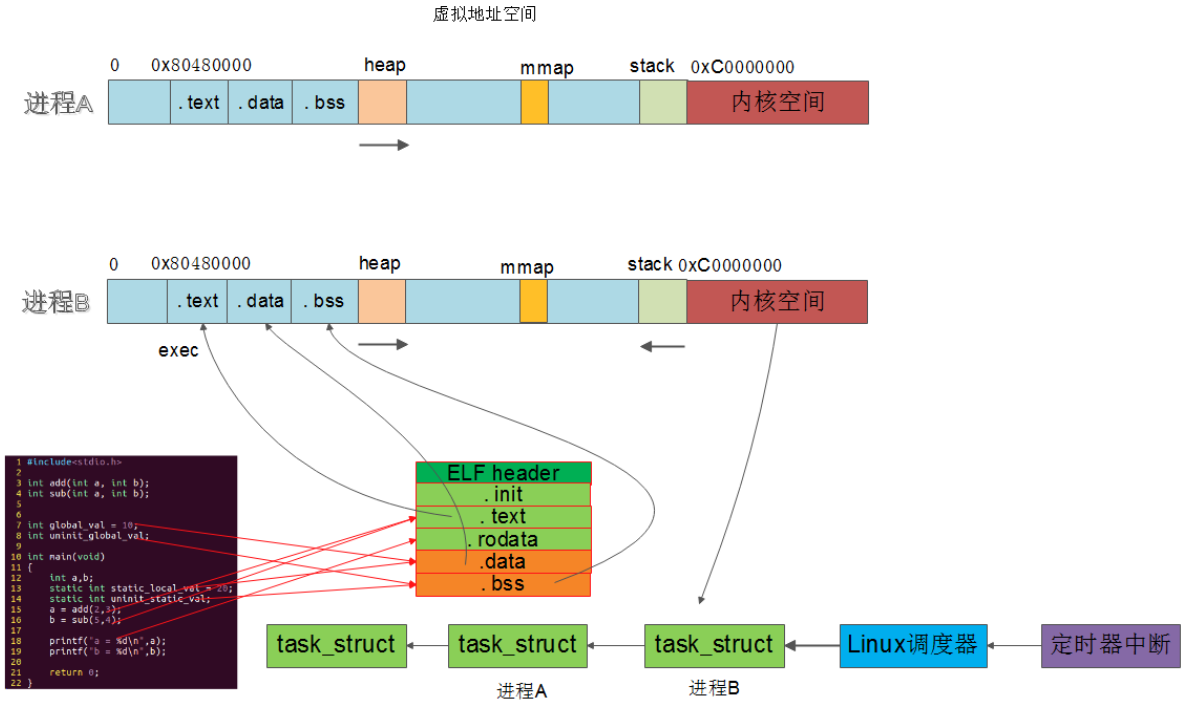
当我们在桌面上点击一个图标，或者在命令行下敲击一个shell命令运行，Linux系统会把这些可执行文件加载到内存，并封装成一个进程，然后才能参与操作系统的调度、执行。那操作系统是如何加载的呢？



首先，我们编写的C语言源代码会编译成一个可执行文件(ELF)。可执行文件分有各种不同的段(section)组成：代码段、数据段、BSS段等。我们C程序中的不同代码会被编译到不同的段中：函数会放到代码段、全局变量、静态局部变量会放到数据段、未初始化的全局变量会放到BSS段中……

加载器在加载程序到内存执行时，分2步走：第一步，会首先使用fork去创建一个子进程，每个进程会有4G的虚拟地址空间。第二步，会从磁盘上软件安装的位置，去读取可执行文件的头部：ELF header，获取各个段的信息，然后分别将不同的段加载到进程空间的不同位置，如上图所示。

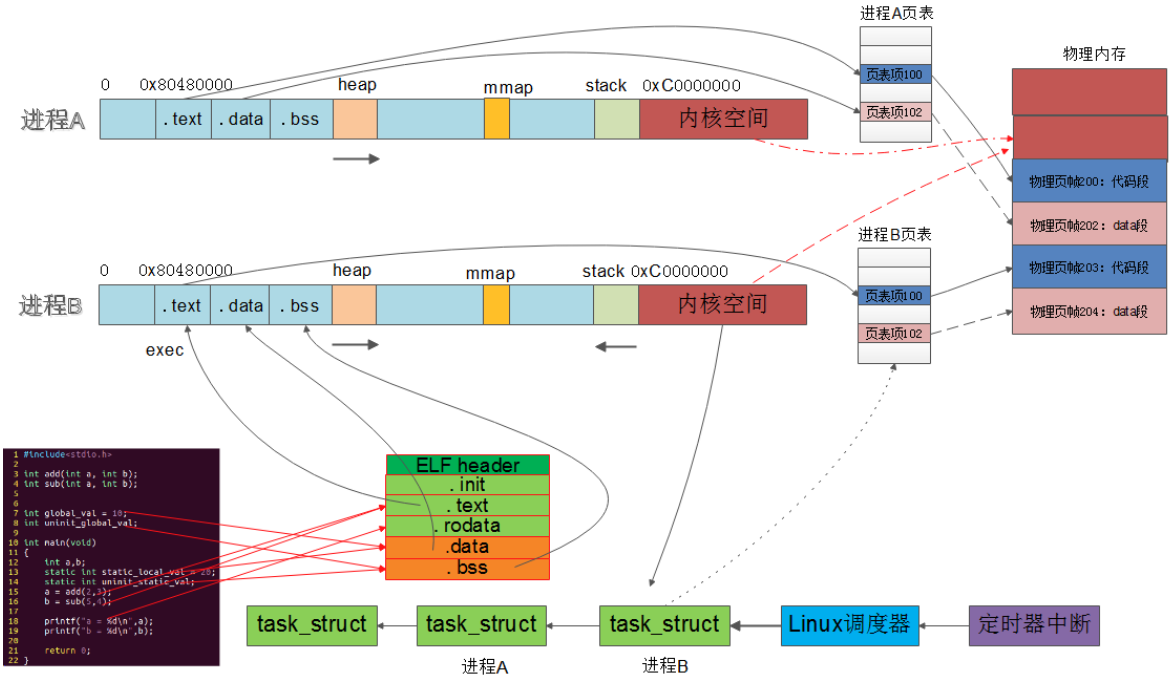
在一个计算机系统中，通常会有多个进程同时运行，每一个进程差不多都是使用这种方式运行的。当运行的进程多了，每个进程都想霸占CPU去运行自己，CPU的资源就不够用了，这个时候操作系统就开始登场了。操作系统扮演一个调度者的角色，协调各个进程轮流占用CPU运行。



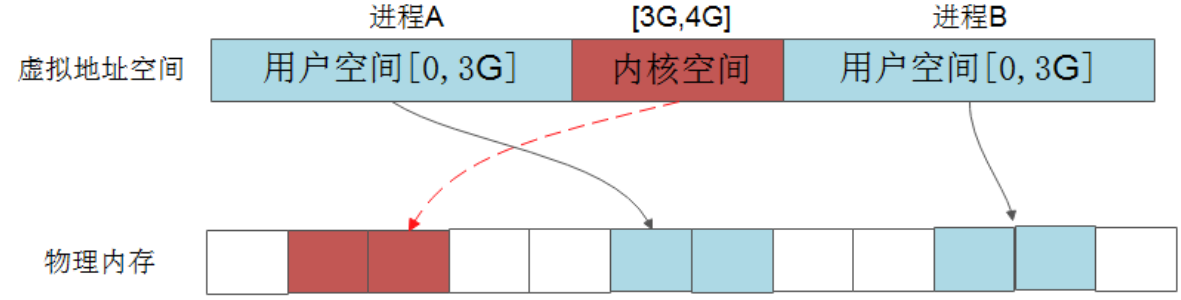
如上图所示，对于用户运行的不同进程，在内核空间，会有一个专门的数据结构来表示：task_struct。这个结构体描述了进程的各种信息，不同的task_sruct结构体通过链表串起来，内核通过链表就可以对这些进程进行管理。操作系统会有一个叫调度器的核心部件，每隔一段时间(一般几毫秒)会有一个定时器中断，Linux调度器就会把正在运行的进程从CPU上赶下来，接着让另一个进程去执行，如此反复，周而复始。

2. 进程的地址空间

每一个进程，都有一个独立的、相同的、4G大小虚拟地址空间，然后通过页表映射，映射到物理内存的不同位置。CPU执行不同的进程时，就会到其对应的物理内存上一条一条地取指令、翻译指令、运行指令。

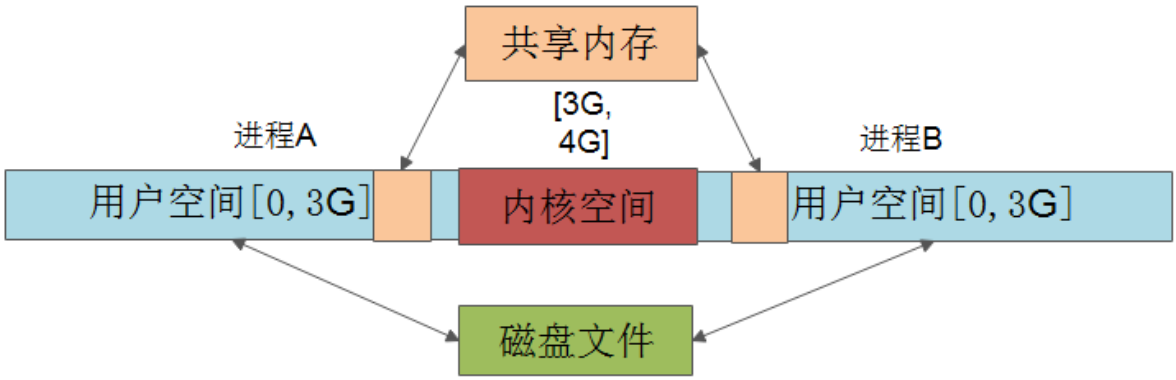


如上图中的进程A和进程B，它们在内存中有相同的4G虚拟地址空间，但是每个进程通过页表映射，就会映射到了物理内存中的不同位置。也就是说，每个进程虽然虚拟地址空间是相同的，但是它们之间是相同隔离的、相互独立的。在每个进程的4G虚拟地址空间中，[0，3G]这段空间是每个进程独有的，而[3G,4G]这段空间是被内核占用的，不同进程的[3G,4G]这段空间都是被内核占用的。内核本身在运行时，在物理内存上会有自己单独的存储空间。



3. Linux进程间通信的三种方法

通过上面的学习我们可以看到，对于每一个运行的进程，它们之间在时空上是相互隔离、相互独立的，如同黑夜和白天，太阳和月亮，永远不会见面。这一幕其实很类似10年前流传于各大网络的《局&长&日&记》：韩%局%长是单核单线程的CPU，每个情人都是一个进程，每个进程都以为自己是独占CPU的，是自己的唯一，每个进程并不知道其它进程的存在。然而现实并非如此：Linux内核把各个进程的信息写在task_struct链表里，韩%局%长把各个进程写到日记里。各个进程之间如果真想通信，还是有方法的，如下图所示。



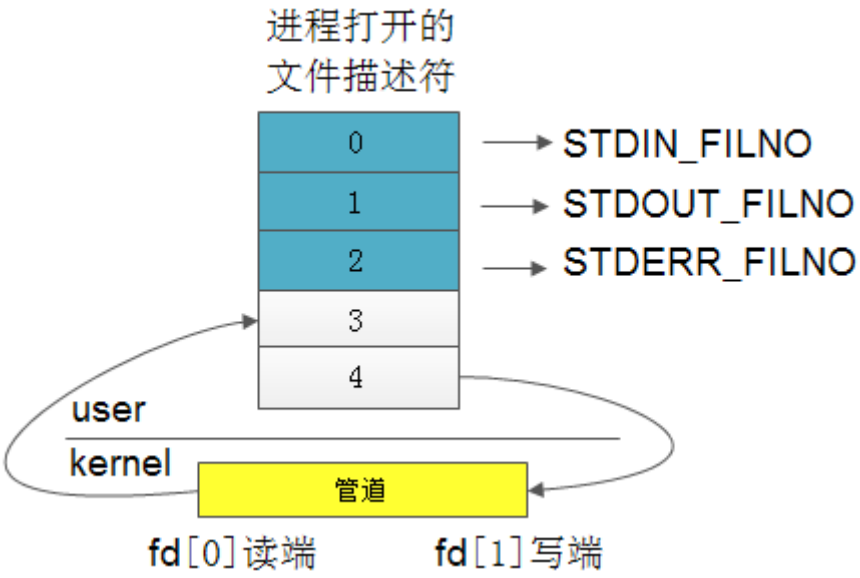
用户空间的每个进程虽说在物理上是相互隔离、相互独立的，但通过内核空间这一共享区域，它们还是可以相互通信的。只要内核愿意、提供一些空间，不同的进程之间就可以往这块内存空间读写数据，来达到通信的目的。磁盘也是公共存储空间，不同进程也可以通过往某个指定的文件读写数据完成进程间的通信。除此之外，不同的进程之间，如果事先商量好，也可以绕过内核，通过内存映射，在物理内存上建立一片共享内存，直接进行通信。

4. 无名管道pipe通信机制

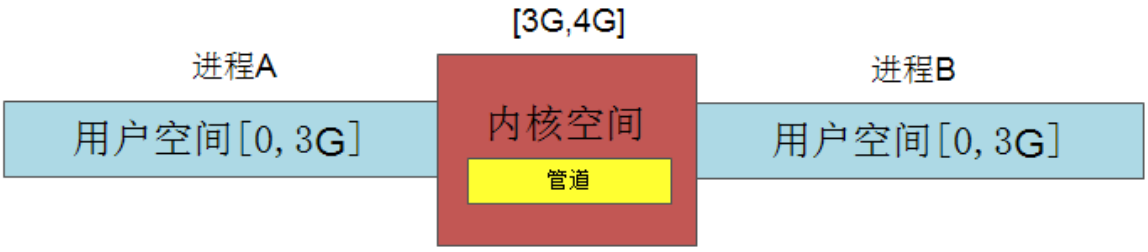
以Linux的无名管道pipe通信机制为例：无名管道常用于有血缘关系的进程之间的通信，我们可以通过pipe系统调用去创建一个管道：

```
int pipe (int  pipefd[2]);
```

该函数会创建一个管道，这个管道有两个文件描述符，一个用来读，一个用来写，不同进程可以通过读写描述符对这个管道进行读写，进而达到通信的目的。



无名管道在内核中的实现其实很简单，就是Linux内核空间的一片缓冲区，通过pipefs机制把它封装成一个文件的形式，留出文件的读写接口：文件描述符给用户空间进程。用户空间的不同进程通过这一对读写描述符就可以对管道进行读写了。



5. 更多的进程间通信工具

除了无名管道外，Linux提供了很多了很多进程间通信的工具可以使用，比如：命名管道FIFO、信号量、消息队列、共享内存、信号signal、socket、Dbus等。不同的IPC工具有各自的优缺点、使用场合。比如无名管道只能用于亲缘关系的进程通信，命名管道PIPE解决了这一局限，支持任意两进程之间的通信；消息队列可以支持有数据格式的通信，共享内存效率最高，但是需要跟信号量、锁等同步机制结合使用；信号主要用于进程间的异步通信，也是唯一的一种异步通信机制。

每一种IPC通信工具，都有自己的优缺点、使用场合和局限，我们只有全面了解和掌握各个IPC工具的使用，知晓其优缺点，才能在实际工作中根据需要，选择合适的通信机制。除了这些标准的POSIX/system V接口定义的IPC工具，Linux系统还扩展了一些自己独特的API，更多教程可以关注：《Linux系统编程》第05期：进程间通信，已在各大平台上传，已经通过淘宝平台预售购买的同学可以直接下载学习了。

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)