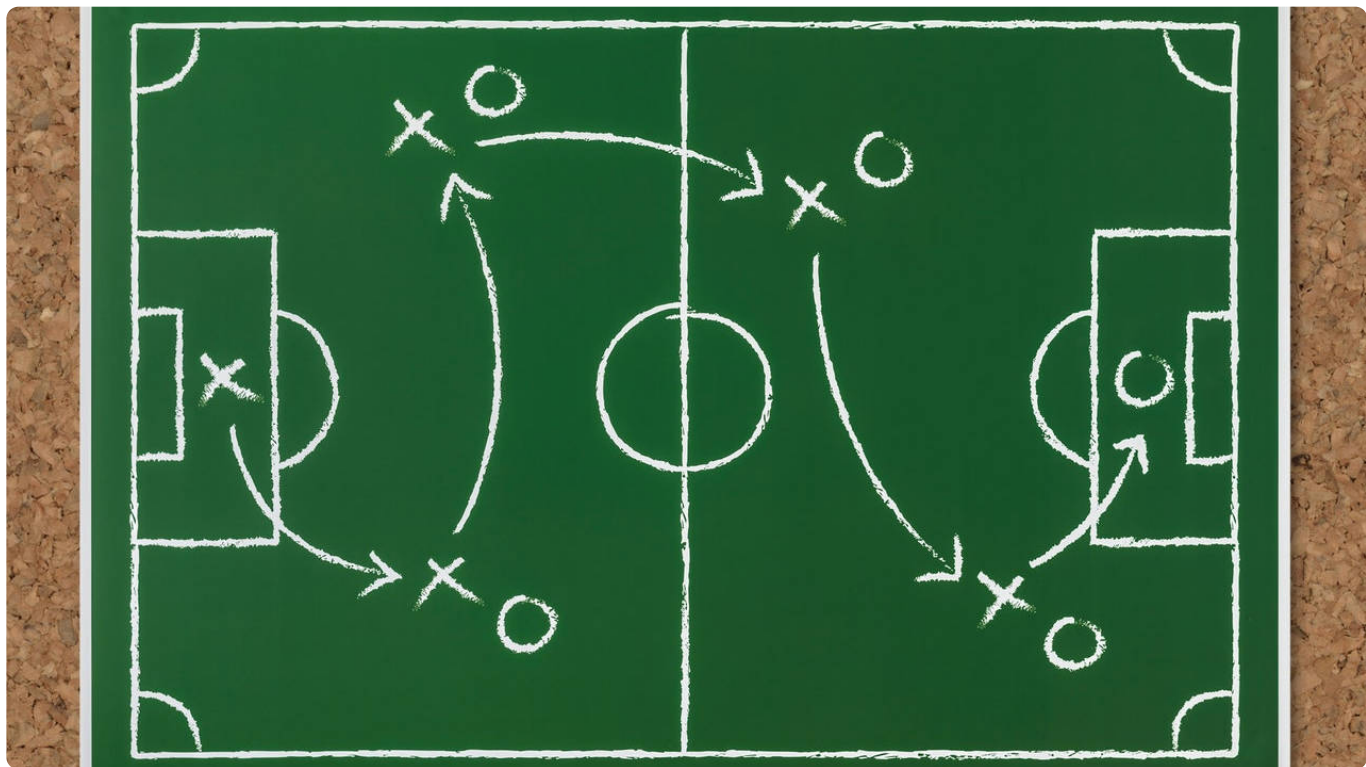


20 | 内存管理（上）：为客户保密，规划进程内存空间布局

2019-05-13 刘超

趣谈Linux操作系统

[进入课程 >](#)



讲述：刘超

时长 12:59 大小 11.90M



平时我们说计算机的“计算”两个字，其实说的就是两方面，第一，进程和线程对于 CPU 的使用；第二，对于内存的管理。所以从这一节开始，我们来看看内存管理的机制。

我之前说把内存管理比喻为一个项目组的“封闭开发的会议室”。很显然，如果不隔离，就会不安全、就会泄密，因而，我们说每个进程应该有自己的内存空间。内存空间都是独立的、相互隔离的。对于每个进程来讲，看起来应该都是独占的。

独享内存空间的原理

之前我只是简单地形容了一下。这一节，我们来深入分析一下，为啥一定要封闭开发呢？

执行一个项目，要依赖于项目执行计划书里的一行一行指令。项目只要按这些指令运行就行了。但是，在运行指令的过程中，免不了要产生一些数据。这些数据要保存在一个地方，这个地方就是内存，也就是我们刚才说的“会议室”。

和会议室一样，**内存都被分成一块一块儿的，都编好了号**。例如 3F-10，就是三楼十号会议室。内存也有这样一个地址。这个地址是实实在在的地址，通过这个地址我们就能够定位到物理内存的位置。

使用这种类型的地址会不会有问题呢？我们的二进制程序，也就是项目执行计划书，都是事先写好的，可以多次运行的。如果里面有个指令是，要把用户输入的数字保存在内存中，那就会有问题。

会产生什么问题呢？我举个例子你就明白了。如果我们使用那个实实在在的地址，3F-10，打开三个相同的程序，都执行到某一步。比方说，打开了三个计算器，用户在这三个程序的界面上分别输入了 10、100、1000。如果内存中的这个位置只能保存一个数，那应该保存哪个呢？这不就冲突了吗？

如果不用这个实实在在的地址，那应该怎么办呢？于是，我们就想出一个办法，那就是**封闭开发**。

每个项目的物理地址对于进程不可见，谁也不能直接访问这个物理地址。操作系统会给进程分配一个虚拟地址。所有进程看到的这个地址都是一样的，里面的内存都是从 0 开始编号。

在程序里面，指令写入的地址是虚拟地址。例如，位置为 10M 的内存区域，操作系统会提供一种机制，将不同进程的虚拟地址和不同内存的物理地址映射起来。

当程序要访问虚拟地址的时候，由内核的数据结构进行转换，转换成不同的物理地址，这样不同的进程运行的时候，写入的是不同的物理地址，这样就不会冲突了。

规划虚拟地址空间

通过以上的原理，我们可以看出，操作系统的内存管理，主要分为三个方面。

第一，物理内存的管理，相当于会议室管理员管理会议室。


第二，虚拟地址的管理，也即在项目组的视角，会议室的虚拟地址应该如何组织。

第三，虚拟地址和物理地址如何映射，也即会议室管理员如果管理映射表。

接下来，我们都会围绕虚拟地址和物理地址展开。这两个概念有点绕，很多时候你可能会犯糊涂：这个地方，我们用的是虚拟地址呢，还是物理地址呢？所以，请你在学习这一章节的时候，时刻问自己这个问题。

我们还是切换到外包公司老板的角度。现在，如果让你规划一下，到底应该怎么管理会议室，你会怎么办？是不是可以先听听项目组的意见，收集一下需求。

于是，你看到了项目组的项目执行计划书是这样一个程序。

 复制代码

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int max_length = 128;
5
6 char * generate(int length){
7     int i;
8     char * buffer = (char*) malloc (length+1);
9     if (buffer == NULL)
10         return NULL;
11     for (i=0; i<length; i++){
12         buffer[i]=rand()%26+'a';
13     }
14     buffer[length]='\0';
15     return buffer;
16 }
17
18 int main(int argc, char *argv[])
19 {
20     int num;
21     char * buffer;
22
23     printf ("Input the string length : ");
24     scanf ("%d", &num);
25
26     if(num > max_length){
27         num = max_length;
28     }
29
30     buffer = generate(num);
31
```

```
32  printf ("Random string is: %s\n",buffer);
33  free (buffer);
34
35  return 0;
36 }
```

这个程序比较简单，就是根据用户输入的整数来生成字符串，最长是 128。由于字符串的长度不是固定的，因而不能提前知道，需要动态地分配内存，使用 malloc 函数。当然用完了需要释放内存，这就要使用 free 函数。

我们来总结一下，**这个简单的程序使用哪些内存的几种方式：**

代码需要放在内存里面；

全局变量，例如 max_length；

常量字符串 "Input the string length : "；

函数栈，例如局部变量 num 是作为参数传给 generate 函数的，这里面涉及了函数调用，局部变量，函数参数等都是保存在函数栈上面的；

堆，malloc 分配的内存存在堆里面；

这里面涉及对 glibc 的调用，所以 glibc 的代码是以 so 文件的形式存在的，也需要放在内存里面。

这就完了吗？还没有呢，别忘了 malloc 会调用系统调用，进入内核，所以这个程序一旦运行起来，内核部分还需要分配内存：

内核的代码要在内存里面；

内核中也有全局变量；

每个进程都要有一个 task_struct；

每个进程还有一个内核栈；

在内核里面也有动态分配的内存；

虚拟地址到物理地址的映射表放在哪里？

竟然收集了这么多的需求，看来做个内存管理还是挺复杂的啊！

我们现在来问一下自己，上面的这些内存里面的数据，应该用虚拟地址访问呢？还是应该用物理地址访问呢？

你可能会说，这很简单嘛。用户态的用虚拟地址访问，内核态的用物理地址访问。其实不是的。你有没有想过，内核里面的代码如果都使用物理地址，就相当于公司里的项目管理部、文档管理部门都可以直接使用实际的地址访问会议室，这对于会议室管理部门来讲，简直是一个“灾难”。因为一旦到了内核，大家对于会议室的访问都脱离了会议室管理部门的控制。

所以，我们应该清楚一件事情，真正能够使用会议室的物理地址的，只有会议室管理部门，所有其他部门的行为涉及访问会议室的，都要统统使用虚拟地址，统统到会议室管理部门哪里转换一道，才能进行统一的控制。

我上面列举出来的，对于内存的访问，用户态的进程使用虚拟地址，这点毫无疑问，内核态的也基本都是使用虚拟地址，只有最后一项容易让人产生疑问。虚拟地址到物理地址的映射表，这个感觉起来是内存管理模块的一部分，这个是“实”是“虚”呢？这个问题先保留，我们暂不讨论，放到内存映射那一节见分晓。

既然都是虚拟地址，我们就先不管映射到物理地址以后是如何布局的，反正现在至少从“虚”的角度来看，这一大片连续的内存空间都是我的了。

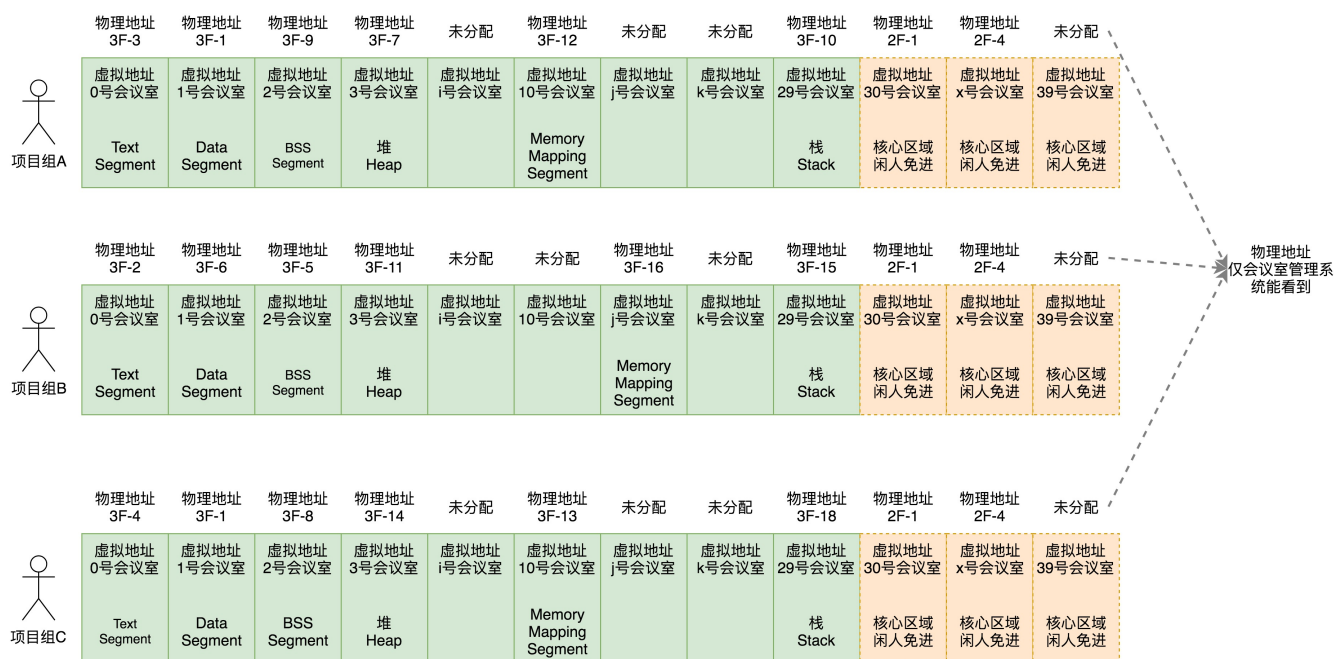
如果是 32 位，有 $2^{32} = 4\text{G}$ 的内存空间都是我的，不管内存是不是真的有 4G。如果是 64 位，在 x86_64 下面，其实只使用了 48 位，那也挺恐怖的。48 位地址长度也就是对应了 256TB 的地址空间。我都没怎么见过 256T 的硬盘，别说是内存了。

现在，你可比世界首富房子还大。虽然是虚拟的。下面你可以尽情地去排列咱们要放的东西。请记住，现在你是站在一个进程的角度去看这个虚拟的空间，不用管其他进程。

首先，这么大的虚拟空间一切二，一部分用来放内核的东西，称为**内核空间**，一部分用来放进程的东西，称为**用户空间**。用户空间在下，在低地址，我们假设就是 0 号到 29 号会议室；内核空间在上，在高地址，我们假设是 30 号到 39 号会议室。这两部分空间的分界线因为 32 位和 64 位的不同而不同，我们这里不深究。

对于普通进程来说，内核空间的那部分虽然虚拟地址在那里，但是不能访问。这就像作为普通员工，你明明知道财务办公室在这个 30 号会议室门里面，但是门上挂着“闲人免进”，

你只能在自己的用户空间里面折腾。



我们从最低位开始排起，先是**Text Segment**、**Data Segment** 和 **BSS Segment**。Text Segment 是存放二进制可执行代码的位置，Data Segment 存放静态常量，BSS Segment 存放未初始化的静态变量。是不是觉得这几个名字很熟悉？没错，咱们前面讲 ELF 格式的时候提到过，在二进制执行文件里面，就有这三个部分。这里就是把二进制执行文件的三个部分加载到内存里面。

接下来是**堆（Heap）段**。堆是往高地址增长的，是用来动态分配内存的区域，malloc 就是在这里面分配的。

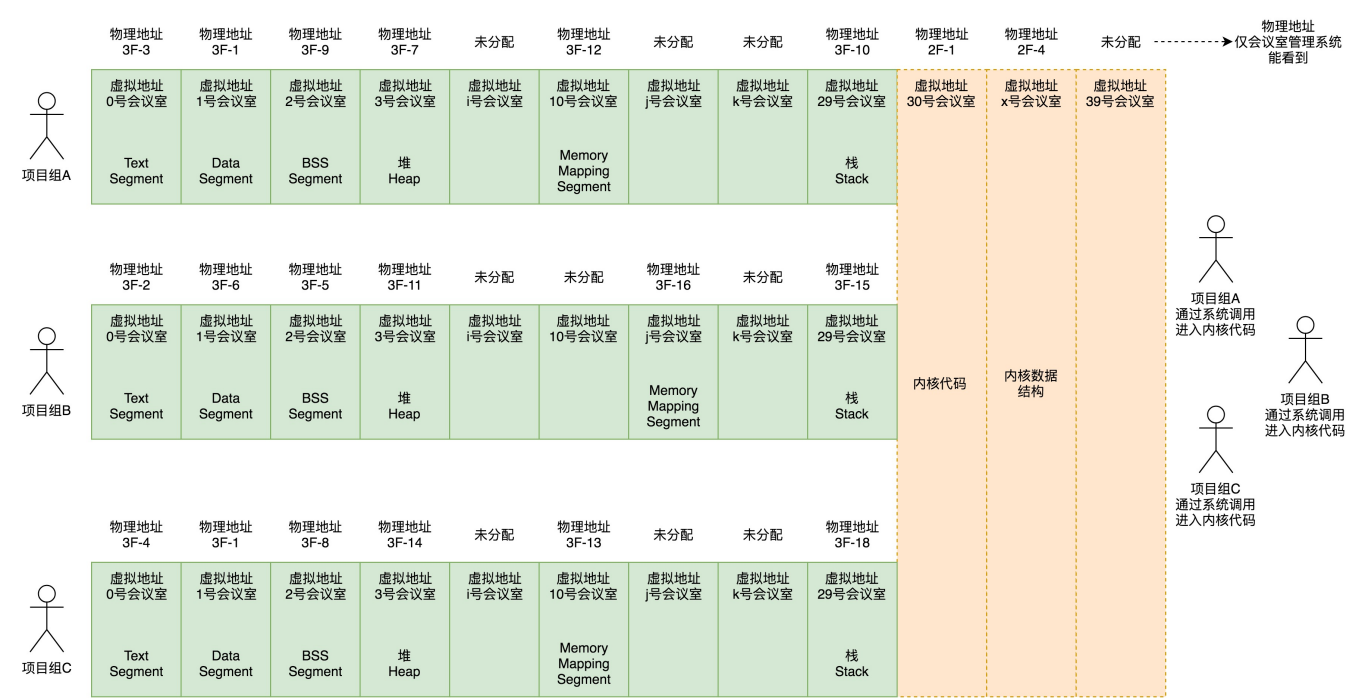
接下来的区域是**Memory Mapping Segment**。这块地址可以用来把文件映射进内存用的，如果二进制的执行文件依赖于某个动态链接库，就是在这个区域里面将 so 文件映射到了内存中。

再下面就是**栈（Stack）地址段**。主线程的函数调用的函数栈就是用这里的。

如果普通进程还想进一步访问内核空间，是没办法的，只能眼巴巴地看着。如果需要进行更高权限的工作，就需要调用系统调用，进入内核。

一旦进入了内核，就换了一副视角。刚才是普通进程的视角，觉着整个空间是它独占的，没有其他进程存在。当然另一个进程也这样认为，因为它们互相看不到对方。这也就是说，不同进程的 0 号到 29 号会议室放的东西都不一样。

但是到了内核里面，无论是从哪个进程进来的，看到的都是同一个内核空间，看到的都是同一个进程列表。虽然内核栈是各用个的，但是如果想知道的话，还是能够知道每个进程的内核栈在哪里的。所以，如果要访问一些公共的数据结构，需要进行锁保护。也就是说，不同的进程进入到内核后，进入的 30 号到 39 号会议室是同一批会议室。



内核的代码访问内核的数据结构，大部分的情况下都是使用虚拟地址的，虽然内核代码权限很大，但是能够使用的虚拟地址范围也只能在内核空间，也即内核代码访问内核数据结构。只能用 30 号到 39 号这些编号，不能用 0 到 29 号，因为这些是被进程空间占用的。而且，进程有很多个。你现在在内核，但是你不知道当前指的 0 号是哪个进程的 0 号。

在内核里面也会有内核的代码，同样有 Text Segment、Data Segment 和 BSS Segment，别忘了咱们讲内核启动的时候，内核代码也是 ELF 格式的。

内核的其他数据结构的分配方式就比较复杂了，这一节我们先不讲。

总结时刻

好了，这一节就到这里了，我们来总结一下。这一节我们讲了为什么要独享内存空间，并且站在老板的角度，设计了虚拟地址空间应该存放的数据。

通过这一节，你应该知道，一个内存管理系统至少应该做三件事情：

- 第一，虚拟内存空间的管理，每个进程看到的是独立的、互不干扰的虚拟地址空间；
- 第二，物理内存的管理，物理内存地址只有内存管理模块能够使用；
- 第三，内存映射，需要将虚拟内存和物理内存映射、关联起来。

课堂练习

这一节我们讲了进程内存空间的布局，请找一下，有没有一个命令可以查看进程内存空间的布局，打印出来看一下，这对我们后面解析非常有帮助。

欢迎留言和我分享你的疑惑和见解，也欢迎你收藏本节内容，反复研读。你也可以把今天的内容分享给你的朋友，和他一起学习、进步。

 极客时间

趣谈 Linux 操作系统

像故事一样的操作系统入门课

刘超

网易杭州研究院
云计算技术部首席架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 19 | 线程的创建：如何执行一个新子项目？

下一篇 21 | 内存管理（下）：为客户保密，项目组独享会议室封闭开发

精选留言 (11)

 写留言



lfn

2019-05-13

👍 11

`cat /proc/$pid/map`

展开 ▾



why

2019-05-14

👍 7

- 内存管理(上)
- 内存管理包含: 物理内存管理; 虚拟内存管理; 两者的映射
- 除了内存管理模块, 其他都使用虚拟地址(包括内核)
- 虚拟内存空间包含: 内核空间(高地址); 用户空间(低地址)
- 用户空间从低到高布局为: 代码段; DATA 段; BSS 段(未初始化静态变量); 堆段; 内存映...

展开 ▾



Linuxer

2019-05-13

👍 2

`pmap pid`

展开 ▾



sam

2019-05-13

👍 1

请问一下, 分析的是linux kernel那个版本?

展开 ▾



雪人

2019-05-13

👍 1

老师, 我想问一下, 所有进程的内核空间是共用一块内存吗? 还有就是, 是不是可以这样理解, 用户空间是负责办事情的, 内核空间是负责管理所有进程的资源, 以及负责与内核一些不公开的资源进行交互的?

展开 ▾



aiter

2019-05-13

👍 1

`cat /proc/22528/maps`

```
00400000-00406000 r-xp 00000000 08:03 2886947 /usr/bin/sleep
00606000-00607000 r--p 00006000 08:03 2886947 /usr/bin/sleep
00607000-00608000 rw-p 00007000 08:03 2886947 /usr/bin/sleep
01ecc000-01eed000 rw-p 00000000 00:00 0 [heap]...
```

展开 ∨



活的潇洒

2019-05-29



配合《深入浅出计算机组成原理》和《Linux性能优化实战》一起学
感觉《趣谈Linux操作系统》难度最大，希望自己能坚持把笔记做到最后
day20笔记: <https://www.cnblogs.com/luoahong/p/10919317.html>

展开 ∨



烈日融雪

2019-05-23



...

```
<?php
if(time()%10==5) {
    echo "I got it".PHP_EOL;
}...
```

展开 ∨



铁皮

2019-05-23



@ CHEN

你的程序有问题，如果sleep在malloc之前调用，那么是看不到[heap]的，是因为还没有程序在heap上分配空间。

如果你在malloc之后调用sleep，就可以看到。



CHEN

2019-05-13



为什么我打印出来的进程内存空间没有堆heap？是因为用的是阿里云的服务器，aiter童鞋是本地安装的虚拟机么？

查看并打印进程空间布局 `cat /proc/$pid/maps`

1 test.c copy老师的示例代码直接编译,或是自己写设置一个中断,sleep(10000);或是getchar();接收键盘输入一个字符...

展开 ∨



有铭

2019-05-13



请问，老师的意思是，内核空间其实是完全共享的吗？大家看到的数据都是相同的，如果修改数据，会影响到所有进程？也就是说，其实内核空间其实只占用一份物理内存？另外，既然都是相同的，所谓“内核栈是各用各的”是啥意思？