第七讲　内核空间和用户空间

从以上几讲我们知道，Linux 简化了分段机制，使得虚拟地址与线性地址总是一致，因此，Linux 的虚拟地址空间也为 0～4G。Linux 内核将这 4G 字节的空间分为两部分。将最高的 1G 字节（从虚拟地址 0xC0000000 到 0xFFFFFFFF），供内核使用，称为“**内核空间**”。而将较低的 3G 字节（从虚拟地址 0x00000000 到 0xBFFFFFFF），供各个进程使用，称为“**用户空间**）。因为每个进程可以通过系统调用进入内核，因此，Linux 内核由系统内的所有进程共享。于是，从具体进程的角度来看，每个进程可以拥有 4G 字节的虚拟空间。图 6.3 给出了进程虚拟空间示意图。

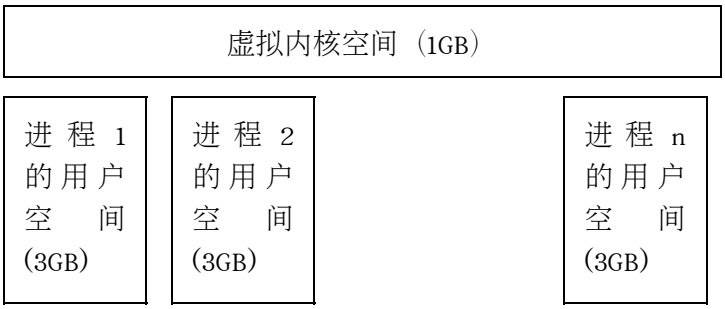


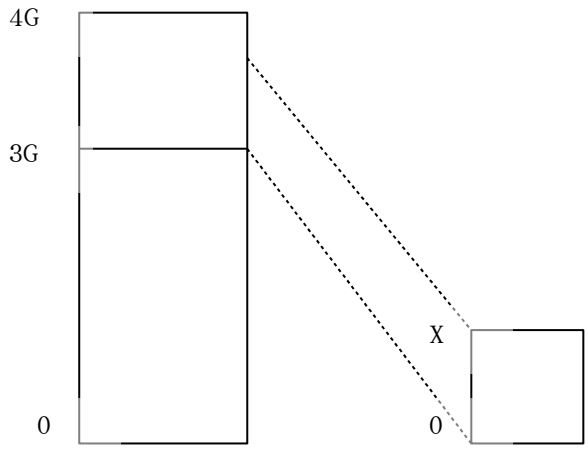
图 6.3　Linux 进程的虚拟空间

Linux 使用两级保护机制：0 级供内核使用，3 级供用户程序使用。从图中可以看出，每个进程有各自的私有用户空间（0～3G），这个空间对系统中的其他进程是不可见的。最高的 1GB 字节虚拟内核空间则为所有进程以及内核所共享。

1．虚拟内核空间到物理空间的映射

内核空间中存放的是内核代码和数据，而进程的用户空间中存放的是用户程序的代码和数据。不管是内核空间还是用户空间，它们都处于虚拟空间中。读者会问，系统启动时，内核的代码和数据不是被装入到物理内存吗？它们为什么也处于虚拟内存中呢？这和编译程序有关，后面我们通过具体讨论就会明白这一点。

虽然内核空间占据了每个虚拟空间中的最高 1GB 字节，但映射到物理内存却总是从最低地址（0x00000000）开始。如图 6.4 所示，对内核空间来说，其地址映射是很简单的线性映射，0xC0000000 就是物理地址与线性地址之间的位移量，在 Linux 代码中就叫做 PAGE_OFFSET。



虚地址空间　　物理地址空间

图 6.4 内核的虚拟地址空间到物理地址空间的映射

我们来看一下在 include/asm/i386/page.h 中对内核空间中地址映射的说明及定义：

```
/*
* This handles the memory map.. We could make this a config
```

```
 * option, but too many people screw it up, and too few need
 * it.
 *
 * A __PAGE_OFFSET of 0xC0000000 means that the kernel has
 * a virtual address space of one gigabyte, which limits the
 * amount of physical memory you can use to about 950MB.
 *
 * If you want more physical memory than this then see the CONFIG_HIGHMEM4G
 * and CONFIG_HIGHMEM64G options in the kernel configuration.
 */

#define __PAGE_OFFSET           (0xC0000000)

......

#define PAGE_OFFSET             ((unsigned long)__PAGE_OFFSET)
#define __pa(x)                 ((unsigned long)(x)-PAGE_OFFSET)
#define __va(x)                 ((void *)((unsigned long)(x)+PAGE_OFFSET))
```

源代码的注释中说明，如果你的物理内存大于 950MB，那么在编译内核时就需要加 CONFIG_HIGHMEM4G 和 CONFIG_HIGHMEM64G 选项，这种情况我们暂不考虑。如果物理内存小于 950MB，则对于内核空间而言，给定一个虚地址 x，其物理地址为“x- PAGE_OFFSET”，给定一个物理地址 x，其虚地址为“x+ PAGE_OFFSET”。

这里再次说明，宏__pa()仅仅把一个内核空间的虚地址映射到物理地址，而决不适用于用户空间，用户空间的地址映射要复杂得多。

2．内核映像

在下面的描述中，我们把内核的代码和数据就叫内核映像（kernel image）。当系统启动时，Linux 内核映像被安装在物理地址 0x00100000 开始的地方，即 1MB 开始的区间(第 1M 留作它用)。然而，在正常运行时， 整个内核映像应该在虚拟内核空间中，因此，连接程序在连接内核映像时，在所有的符号地址上加一个偏移量 PAGE_OFFSET，这样，内核映像在内核空间的起始地址就为 0xC0100000。

例如，进程的页目录 PGD（属于内核数据结构）就处于内核空间中。在进程切换时，

要将寄存器 CR3 设置成指向新进程的页目录 PGD，而该目录的起始地址在内核空间中是虚地址，但 CR3 所需要的是物理地址，这时候就要用__pa()进行地址转换。在 mm_context.h 中就有这么一行语句：

asm volatile(“mov1 %0,%%cr3”：:” r” (__pa(next->pgd));

这是一行嵌入式汇编代码，其含义是将下一个进程的页目录起始地址 next_pgd，通过__pa()转换成物理地址，存放在某个寄存器中，然后用 mov 指令将其写入 CR3 寄存器中。经过这行语句的处理，CR3 就指向新进程 next 的页目录表 PGD 了。