Linux 0.11 内核中内存地址空间映射

gohigh@sh163.net

2004-10-22

Linux 0.11 操作系统从机器复位到进入具有分页功能的 32 位保护运行方式，整个过程需要我们对内存空间的划分和使用有一个清晰的了解。这里我们对进入多任务的 32 位保护运行方式后机器的状态进行详细说明。

(1) 设定

对于 Linux 0.11 内核运行的硬件环境，我们假定使用的是一般的普通 PC 机，其物理内存分配情况见图 2-9。我们假定机器共有 16MB 的物理内存。从物理内存地址 0xA0000（640KB）到 0x100000（1MB）是显示卡和 ROM 占用的位置。

(2) 多任务和保护

Linux 0.11 操作系统使用了 CPU 的 0 和 3 两个保护级。内核代码本身会由系统中的所有任务共享。而每个任务则都有自己的代码和数据区，这两个区域保存于局部地址空间，因此系统中的其他任务是看不见的（不能访问的）。而内核代码和数据是由所有任务共享的，因此它保存在全局地址空间中。图 1 给出了这种结构的示意图。图中同心圆代表 CPU 的保护级别（保护层），这里仅使用了 CPU 的 0 级和 3 级。而径向射线则用来区分系统中的各个任务。每条径向射线指出了一个任务的边界。除了每个任务虚拟地址空间的全局地址区域，任务 1 中的地址与任务 2 中相同地址处是无关的。

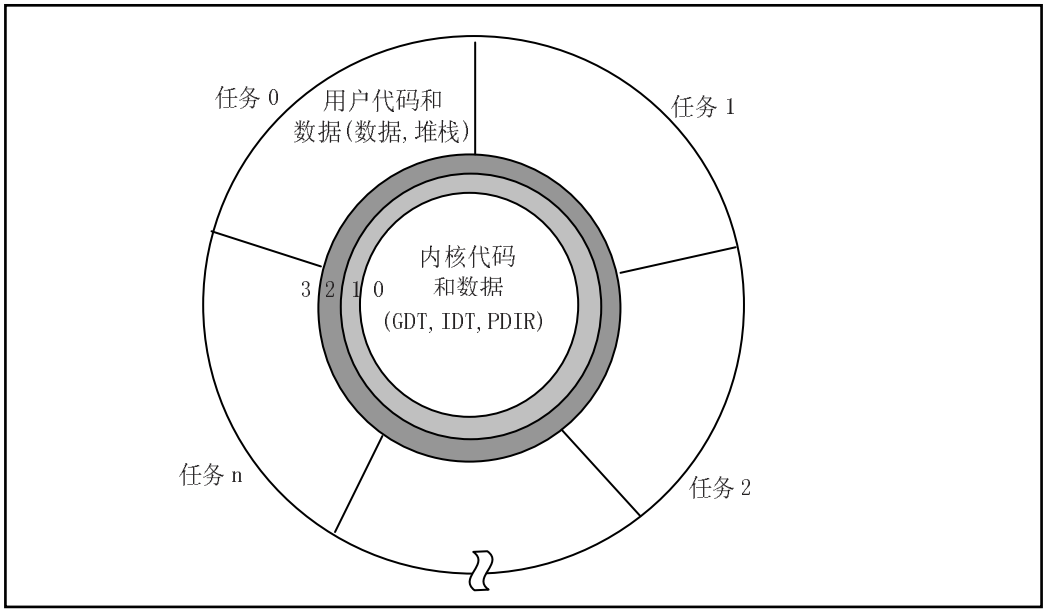

图 1 多任务系统

(3) 虚拟地址空间

虚拟地址（Virtual address）是程序产生的由段和偏移值两部分构成的地址。由于这两部构成的地址并没有直接用来访问物理内存，而是需要通过分段地址变换机制处理或映射后才对应到物理内存地址上，因此称之为虚拟地址。

图 2 显示出了每个任务的虚拟地址空间。虚拟地址空间由 GDT 映射的全局地址空间和 LDT 映射的局

部地址空间组成。对于第一个任务来讲全局地址空间含有 5 个段：

1. GDT 段本身。它含有在全局地址空间中定义其他 4 个段的描述符。
2. 第 1 个任务的 TSS 段。当系统中运行多个任务时，必须为每个任务定义一个 TSS 段。
3. 操作系统内核的代码段。
4. 操作系统内核的数据段。
5. 第 1 个任务的 LDT 段，用于在局部地址空间中映射各段。当系统中运行多个任务时，必须为每个任务定义一个 LDT 段。

局部地址空间含有两个段：

1. 用户代码段。
2. 用户数据段。该段中含有程序数据和堆栈。

而应用程序只能看见一个含有单个数据段和代码段的单段模型。

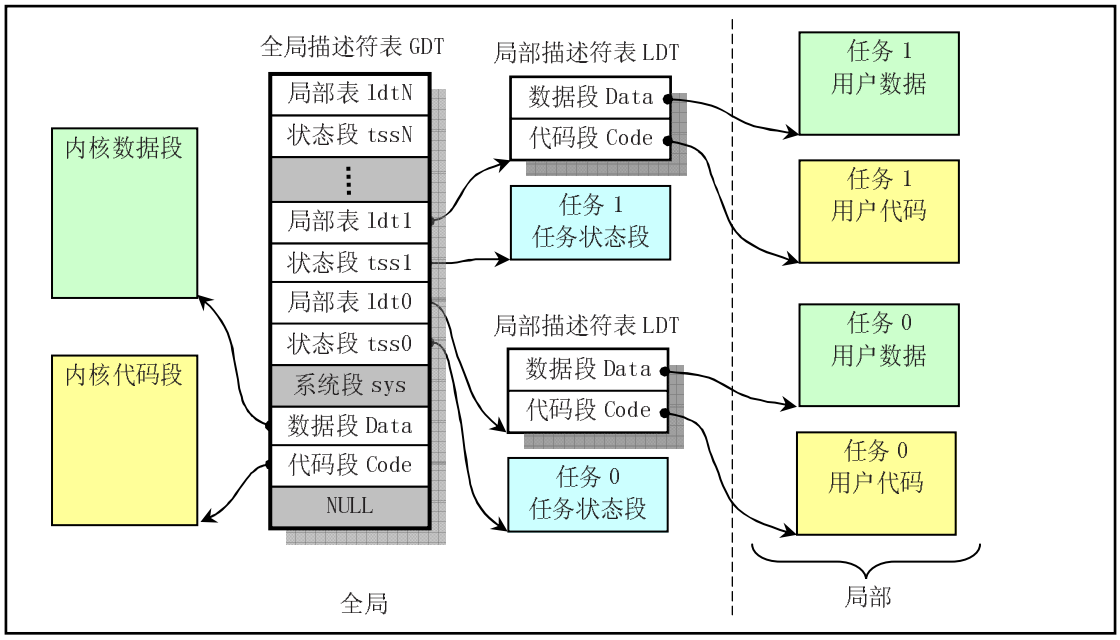

图 2 虚拟地址空间

(4) 线性地址空间

线性地址空间分配示意图见图 3 所示。在线性地址空间中有如下描述的几个段。

1. 内核代码段和数据段。它们都被定义为从线性地址 0 开始到地址 0xFFFFFF 共 16MB 长度的段，因此内核代码段与内核数据段的范围相互重叠。在该范围中含有内核所有代码、内核段表（GDT、IDT、TSS）、页目录表和内核二级页表、内核局部数据以及内核临时堆栈（将被用作第 1 个任务的用户堆栈）。
2. 第 1 个任务（任务 0，即 idle 任务）的代码段和数据段。Linux 0.11 中的第 1 个任务比较特殊。它的代码和数据段包含在内核代码和数据段中，并且所占范围也完全重叠。范围是从线性地址 0 开始的 640KB 空间。在 Linux 0.11 系统中，所有任务的代码段和数据段都被设置成在线性地址空间中完全重叠的范围。
3. 第 2 个任务（任务 1，即 init 任务）的代码段和数据段。它占用线性地址空间中的 64MB 到 128MB 范围，两个段重叠，长度都是 64MB。若任务 0 用 nr=0 表示，则任务 1 在线性地址空间中的起始位置即为 nr*64MB=1*64MB=64MB，任务 2 的代码和数据段的起始位置则为 rn*64MB=2*64MB=128MB，其他任务依次类推。

内核代码段和数据段占用线性地址空间的开始 16MB 部分，并且代码和数据段完全相重叠。而第 1 个任务（任务 0）是包含在内核代码和数据中的，因此该任务所占用的线性地址空间范围比较特殊。任务 0 的代码段和数据段的长度是从线性地址 0 开始的 640KB 范围，其代码和数据段也是完全重叠的，并且与内

核代码段和数据段有重叠的部分。因此实际上任务 0 的线性地址范围与图中所示略有差异。

把内核代码段和数据段选择在线性地址开始的范围处，是为了让内核代码和数据在线性地址和物理地址上相同。这样在系统启动后进入保护模式前内核被加载的位置与进入保护模式后的位置相同，既便于调试，也便于分页机制的实现。

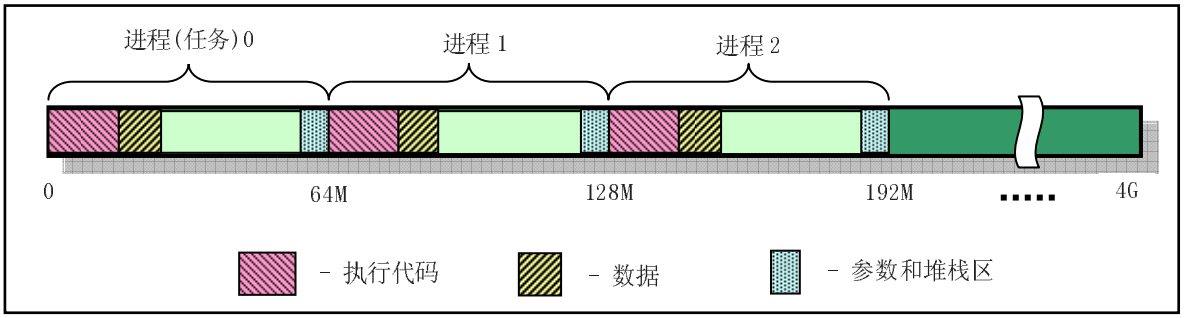

图 3 任务在线性地址空间中的位置

(5) 页映射

线性地址空间-- 由分段机制变换后的地址空间 -- 在由页表进行映射。图中显示出线性地址空间被分成一个个 4MB 大小的区域。4MB 是一个页表所能管理的范围(一页内存含有 1024 个页表项)。在 Linux 0.11 内核初始化过程中（head.s 程序中），从页目录表开始处为内核设置了 4 个页目录项，指向 4 个页表，共可管理 16MB 的线性地址范围，并一一对应地映射到 16MB 物理内存上。这使得内核代码可以访问 16MB 物理内存的任何地方。

对于任务 0，因为其代码和数据在内核区域中，其线性地址已经作过映射，因此不需要再为其设置页目录和页表项。当系统创建任务 1 时，由于它所占有的虚拟地址空间不同（从 64MB 开始），因此需要为它复制父进程（任务 0）的页目录项和页表项。对于其他随后产生的任务也作同样的处理。

(6) 物理地址

对物理地址空间的使用分配方式见图 4 所示。默认情况下，Linux 0.11 内核可管理 16MB 的物理内存，共有 4096 个物理页面（页帧），每个页面 4KB。可以看出：①内核代码和数据段区域在线性地址空间和物理地址空间中是一样的。这样设置可以大大简化内核的初始化操作。②GDT 和 IDT 在内核数据段中，因此它们的线性地址也同样等于它们的物理地址。否则的话，在进入保护模式和开启分页机制时这两个表就需要被重新建立或移动位置，描述符也需要重新加载。③除任务 0 以外的其他任务所需要的物理内存页面与线性地址中的不同，因此内核需要动态地在主内存区中为它们作映射操作，动态地建立页目录项和页表项。

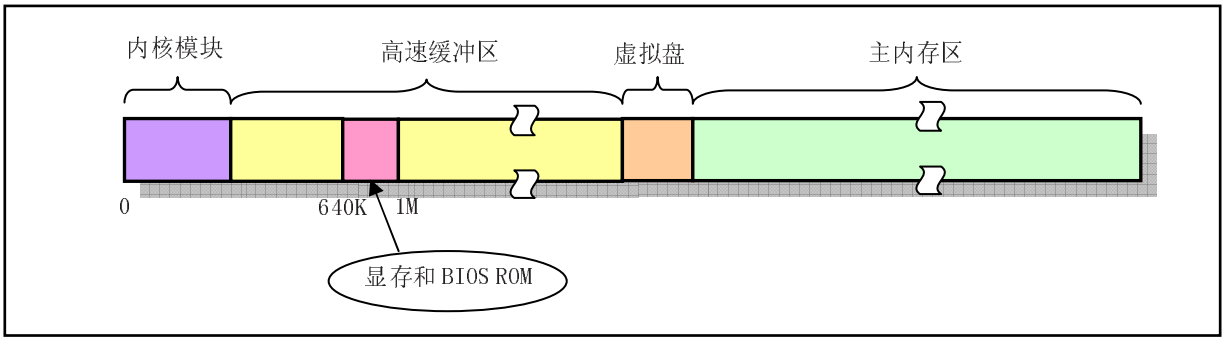

图 4 物理内存空间

（完）