C标准定义）。那么malloc是如何实现的就被推到了POSIX标准之外了。在一些圈子里，这种方法被认为是推卸责任。

实际上，许多Linux系统有管理内存的系统调用。最常见的列在了图10-14中。brk通过给出数据段之外的第一个字节地址来指定数据段的大小。如果新值比原来的要大，那么数据段变大；反之，数据段缩减。

系统调用	描述
s=brk (addr)	改变数据段大小
a=mmap (addr,len,prot,flags,fd,offset)	映射文件
s=unmap (addr,len)	取消映射文件

图10-14 跟内存管理相关的一些系统调用。若遇到错误则返回码s为－1；a和addr是内存地址，len是长度，prot是控制保护，flags是混杂位串，fd是文件描述符，offset是文件偏移

mmap和munmap系统调用控制内存映射文件。mmap的第一个参数，addr，决定文件被映射的地址。它必须是页大小的倍数。如果这个参数是0，系统确定地址并且返回到a中。第二个参数len指示要映射的字节数。它也必须是页大小的整数倍。第三个参数，prot，确定对映射文件的保护。它可以标记为可读、可写、可执行或者三者的组合。第四个参数，flags，控制文件是私有的还是共享的以及addr是一个需求还是仅仅是一个提示。第五个参数，fd，是要映射的文件的描述符。只有打开的文件是可以被映射的，因此为了映射一个文件，首先必须要打开它。最后，offset告诉从文件中的什么位置开始映射。并不一定要从第0个字节开始映射，任何页面边界都是可以的。

另一个调用，unmap，移除一个被映射的文件。如果仅仅是文件的一部分撤销映射，那么其他部分仍然保持映射。

10.4.3 Linux中内存管理的实现

32位机器上的每个Linux进程通常有3GB的虚拟地址空间，还有1GB留给其页表和其他内核数据。在用户态下运行时，内核的1GB是不可见的，但是当进程陷入到内核时是可以访问的。内核内存通常驻留在低端物理内存中，但是被映射到每个进程虚拟地址空间顶部的1GB中，在地址0xC0000000和0xFFFFFFFF（3～4GB）之间。当进程创建的时候，进程地址空间被创建，并且当发生一个exec系统调用时被重写。

为了允许多个进程共享物理内存，Linux监视物理内存的使用，在用户进程或者内核构件需要时分配更多的内存，把物理内存动态映射到不同进程的地址空间中去，把程序的可执行体、文件和其他状态信息移入移出内存来高效地利用平台资源并且保障程序执行的进展性。本章的剩余部分描述了在Linux内核中负责这些操作的各种机制的实现。

1. 物理内存管理

在许多系统中由于异构硬件限制，并不是所有的物理内存都能被相同地对待，尤其是对于I/O和虚拟内存。Linux区分三种内存区域（zone）：

1) ZONE_DMA：可以用来DMA操作的页。

2) ZONE_NORMAL：正常规则映射的页。

3) ZONE_HIGHMEM：高内存地址的页，并不永久性映射。

内存区域的确切边界和布局是硬件体系结构相关的。在x86硬件上，一些设备只能在起始的16MB地址空间进行DMA操作，因此ZONE_DMA就在0～16MB的范围内。此外，硬件也不能直接映射896MB以上的内存地址，因此ZONE_HIGHMEM就是高于该标记的任何地址。ZONE_NORMAL是介于其中的任何地址。因此在x86平台上，Linux地址空间的起始896MB是直接映射的，而内核地址空间的剩余128MB是用来访问高地址内存区域的。内核为每个内存区域维护一个zone数据结构，并且可以分别在三个区域上执行内存分配。

Linux的内存由三部分组成。前两部分是内核和内存映射，被“钉”在内存中（页面从来不换出）。内存的其他部分被划分成页框，每一个页框都可以包含一个代码、数据或者栈页面，一个页表页面，或者在空闲列表中。

内核维护内存的一个映射，该映射包含了所有系统物理内存使用情况的信息，比如区域、空闲页框