**Linux操作系统储存器管理**

**引言：**

Linux存储管理的主要功能包括如下三方面：

* 虚拟存储器功能：为每个进程提供4GB的虚拟内存空间
* 存储保护：各个进程的虚拟存储空间彼此独立、相互隔离
* 存储共享：进程之间可以共享存储器内存(Main Memory或Primary Memory或Real Memory)也称主存是指CPU能直接存取指令和数据的存储器。 磁盘、 磁鼓和磁带等存储器， 一般称为外存或辅存（Secondary Storage）。

**正文：**

Linux是为多用户多任务设计的实时操作系统, 所以存储资源要被多个进程有效共享，且由于程序规模的不断膨胀，要求的内存空间比从前大得多，所以 Linux内存管理的设计充分利用了计算机系统所提供的虚拟存储技术，真正实现了虚拟存储器管理。Linux的内存管理主要体现在对虚拟内存的管理上，为此我们可以把Linux虚拟内存管理功能概括为以下几点：大地址空间、进程保护、内存映射、公平的物理内存分配、共享虚拟内存等几方面。

1. **Linux虚拟内存实现结构**

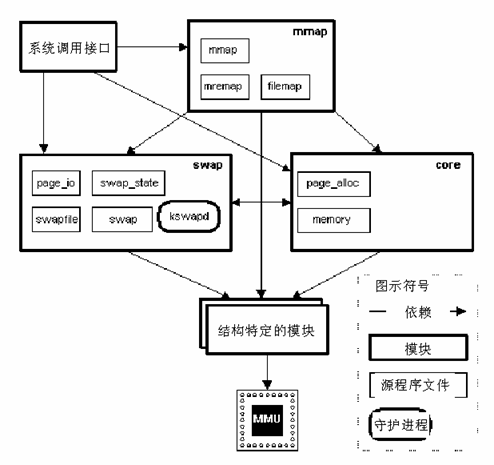


图1 Linux虚存的实现结构

从图中可看到实现虚拟内存的组成模块，其实现的原代码大部分放在/mm目录下。

1). 内存映射模块(mmap)—负责把磁盘文件的逻辑地址映射到虚拟地址，以及把虚拟地址映射到物理地址。

2).交换模块（swap）—负责控制内存内容的换入和换出，它通过交换机制，使得在物理内存的页面（RAM页）中保留有效的页 ，即从主存中淘汰最近没被访问的页，保存近来访问过的页。

3).核心内存管理模块（core)—负责核心内存管理功能，即对页的分配、回收、释放及请页处理等，这些功能将被别的内核子系统（如文件系统）使用。

4). 结构特定的模块—负责给各种硬件平台提供通用接口，这个模块通过执行命令来改变硬件MMU的虚拟地址映射，并在发生页错误时，提供了公用的方法来通知别的内核子系统。这个模块是实现虚拟内存的物理基础。

1. **Linux内存的分配与回收**

1）内存分配：

1、当收到内存分配请求时，系统照buddy算法，根据请求的页面数在free\_area[ ]数组对应的空闲页面组中搜索。

2、若请求的页面不是2的整数次幂，则按照稍大于请求数的2的整数次幂的值搜索对应的页面块组。例如请求2个页面时，则搜索2页面块组，请求3个页面时，就搜索4页面块组。

3、当搜索对应的页块组，而没有可利用的空闲页块时，再搜索更大一些的页块组，例如当查找4页块组没有可利用的空闲页块时，就再搜索8页块组。

4、在找到可利用的空闲页块后，就按照请求的页面块数分配所需的页面。

5、当某一空闲页面块被分配后，若仍有剩余的空闲页面，则根据剩余页面的大小把它们加入相应的空闲页块组中。如图1.4所示：

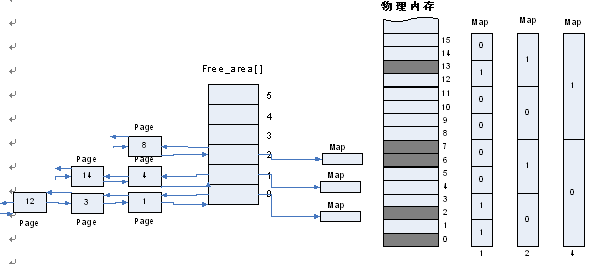


图1.1 Linux虚存的实现结构

2）内存回收：

1、当释放内存页面时，系统将这些页面回收并作为空闲页面看待。

2、然后检查是否存在与这些页面相邻的其它空闲页块，若存在，则把它们合并为一个连续的空闲区。

3、再按照上述buddy算法划分不同的页面块，并加到相应的页面块组中。

1. **Linux物理内存空间的管理**

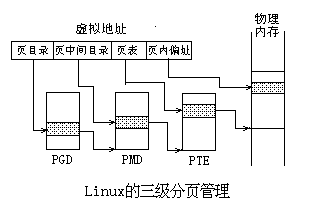
基于物理内存在内核空间中的映射原理，物理内存的管理方式也有所不同。内核中物理内存的管理机制主要有bootmem分配器、伙伴算法，slab高速缓存和vmalloc机制。其中伙伴算法和slab高速缓存都在物理内存映射区（Noamal分配物理内存，而vmalloc机制则在高端内存映射区分配物理内存：

* **bootmem分配器**：是系统启动初期的内存分配方式，在伙伴系统、slab系统建立前内存都是利用bootmem分配器来分配的，只对低于 896M 的物理内存进行分配。伙伴系统框架建立起来后，bootmem会过度到伙伴系统，bootmem大致思想就是收集内存中的可用内存，然后建立bit位图，然后需要的内存从这些空闲内存中分配，分配了就标记占用，当然这种分配方式很低效，但是由于只占用启动阶段很少一部分，所以也大可接受了。
* **伙伴算法：**负责大块连续物理内存的分配和释放，以页框为基本单位。该机制可以避免外部碎片。
* **Slab高速缓存:**负责小块物理内存(小于一个页框)的分配，并且它也作为高速缓存，主要针对内核中经常分配并释放的对象。
* **vmalloc机制**：使得内核可通过连续的线性地址来访问非连续的物理页框，这样可以最大限度的使用高端物理内存。

1. **linux的分页存储管理**
2. **Linux的三级分页结构**
3. **概述**

页表是分页系统中从线性地址向物理地址转换不可缺少的数据结构，而且它使用的频率较高。页表必须存放在物理存储器中。若虚存空间有4GB，按4KB页面划分页表可以有1M页。若采用一级页表机制，页表有1M个表项，每个表项4字节，这个页面就要占用4MB的内存空间。由于系统中每个进程都有自己的页表，如果每个页表占用4MB，对于多个进程而言就要占去大量的物理内存，这是不现实的。Linux采用了一种同时适用于32位和64位系统的普通分页模型。两级页表对32位系统来说已经足够了，但64位系统需要更多数量的分页级别。直到2.6.10版本，Linux采用三级分页的模型。

Linux三级分页管理把虚拟地址分成四个位段：页目录PGD（PaGe Directory）、页中间目录PMD（Page Middle Directory）、页表PTE（Page TablE）、页内偏址。

****

1. **页全局目录（即页目录）**

页全局目录表，最多可包含1024个页目录项，每个页目录项为4个字节，算起来正好一个页面，结构如图所示：



1. **页表**

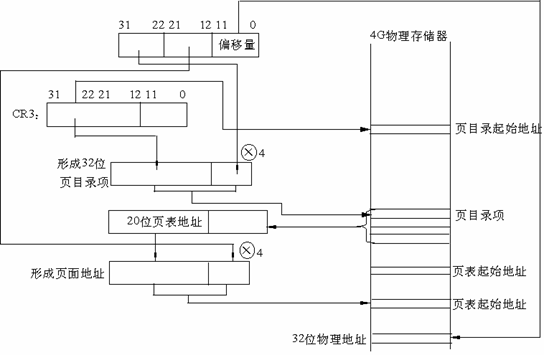
80386的每个页目录项指向一个页表，页表最多含有1024个页面项，每项4个字节，包含页面的起始地址和有关该页面的信息。页面的起始地址也是4K的整数倍，所以页面的低12位也留作它用，如图所示。



第31~12位是20位物理页面地址，除第6位外第0～5位及9~11位的用途和页目录项一样，第6位是页表项独有的，当对涉及的页面进行写操作时，D位被置1。4GB的存储器只有一个页目录，它最多有1024个页目录项，每个页目录项又含有1024个页面项，因此，存储器一共可以分成1024×1024=1M个页面。由于每个页面为4K个字节，所以，存储器的大小正好最多为4GB。

1. **线性地址到物理地址的转换**

在两级页表时，当访问一个操作单元时，如何由分段结构确定的32位线性地址通过分页操作转化成32位物理地址呢？过程如图所示。

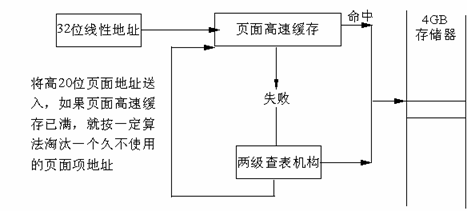


**①第一步**，CR3包含着页目录的起始地址，用32位线性地址的最高10位A31~A22作为页目录的页目录项的索引，将它乘以4，与CR3中的页目录的起始地址相加，形成相应页目录项的地址。

**②第二步**，从指定的地址中取出32位页目录项，它的低12位为0，这32位是页表的起始地址。用32位线性地址中的A21~A12位作为页表中的页面的索引，将它乘以4，与页表的起始地址相加，形成32位页表项地址。

**③第三步**，从指定的地址中取出32位页面地址，将A11~A0作为相对于页面地址的偏移量，与32位页面地址相加，形成32位物理地址。

1. **页面高速缓存**



由于在分页情况下，每次存储器访问都要存取两级页表，这就大大降低了访问速度。所以，为了提高速度，在386中设置一个最近存取页面的高速缓存硬件机制，它**自动保持32项处理器最近使用的页面地址**，因此，可以覆盖128K字节的存储器地址。当进行存储器访问时，先检查要访问的页面是否在高速缓存中，如果在，就不必经过两级访问了，如果不在，再进行两级访问。平均来说，页面高速缓存大约有98%的命中率，也就是说每次访问存储器时，只有2%的情况必须访问两级分页机构。这就大大加快了速度，页面高速缓存的作用如上图所示。

1. **Linux缺页中断处理**

linux在执行一个程序时，虽然已经为程序分配了虚拟地址空间，但只将很少的一部分装入物理内存，所以必然会出现需要访问的虚拟地址空间中某个页面不在内存的情况，此时虚拟存储硬件会触发页故障并报告故障原因。系统中有三种情况会产生页故障：

（1）一是程序出现错误，例如读写一个非法的虚拟地址（如超出虚拟地址空间），此时将产生页故障，Linux将向进程发送SIGSEGV信号终止进程的运行。

（2）第二种情况是，要访问的虚拟地址被写保护，即保护错误，这时，操作系统必须判断：如果是用户进程正在写当前进程的地址空间，则发SIGSEGV信号并终止进程的运行：如果错误发生在一旧的共享页上时，则处理方法有所不同，也就是要对这一共享页进行复制，这就是我们所说的写时复制(Copy On Wrlte简称COW)技术，linux将完成以下工作：

* 使用get\_free\_page()函数重新分配一页；
* 复制发生保护错误页面的内容到新页；
* 修改mem\_map数组反映物理页的变化；
* 重置页目录和页表；
* 刷新缓存。

（3）最后一种情况是，虚拟地址有效，但其所对应的页当前不在物理内存中，即缺页错误，这时，操作系统必须从磁盘映象或交换文件（该页被换出）中将其装入物理内存。

1. **参考文献**

[1] 刘文峰.嵌入式Linux操作系统的研究[J].浙江大学学报，2004，4：310-316.

[2] 张荣亮.Linux操作系统内核分析与研究[D].江西：江西师范大学，2007.

[3] 林文芳，王莉.基于Linux平台的虚拟化研究[J].网络安全技术与应用，2019，09：13-14.

[4] 杜传业.嵌入式Linux内核解析[D].河北：河北工业大学，2007.

[5] 赖娟.Linux内核分析及实时性改造[D].成都：电子科技大学，2007.