专题6-Linux内核子系统

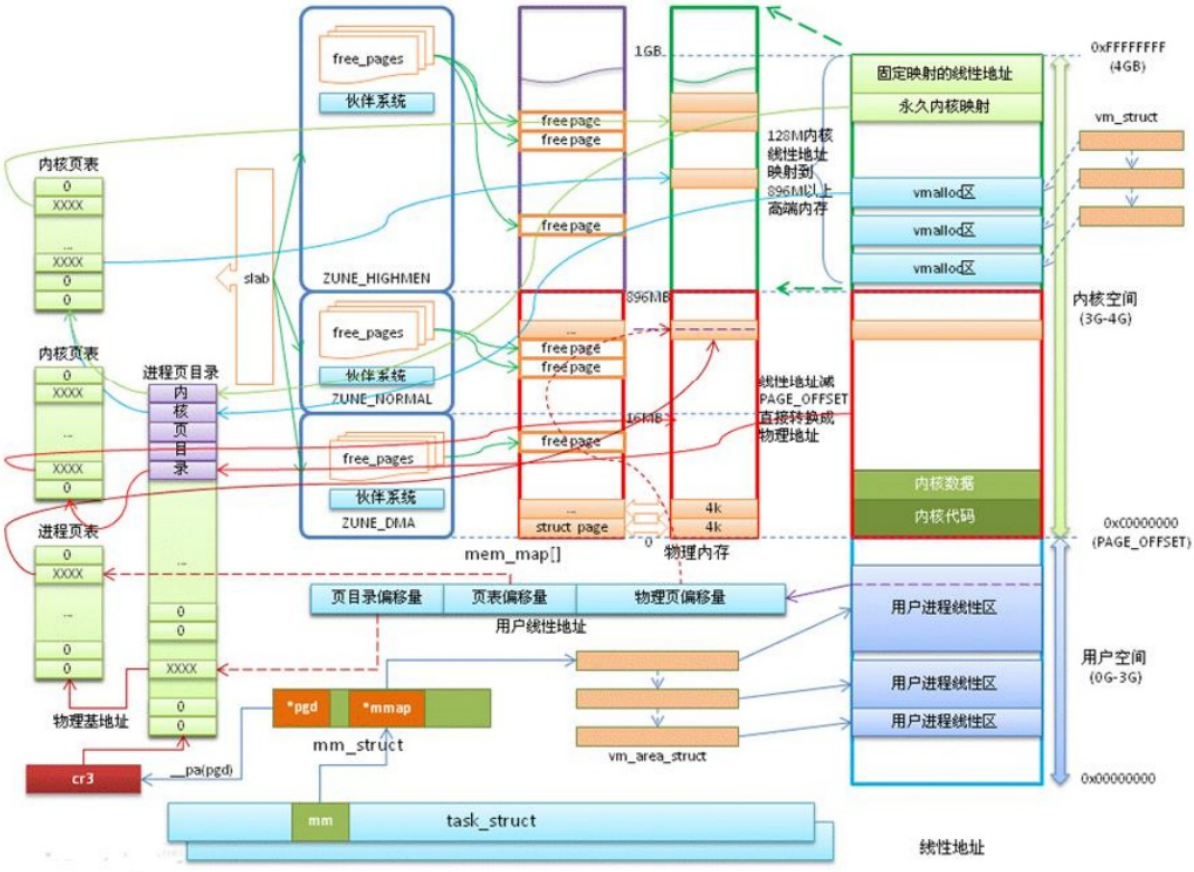
一、Linux内存管理子系统

Linux内存管理子系统职能：

（1）、虚拟地址与物理地址的映射。

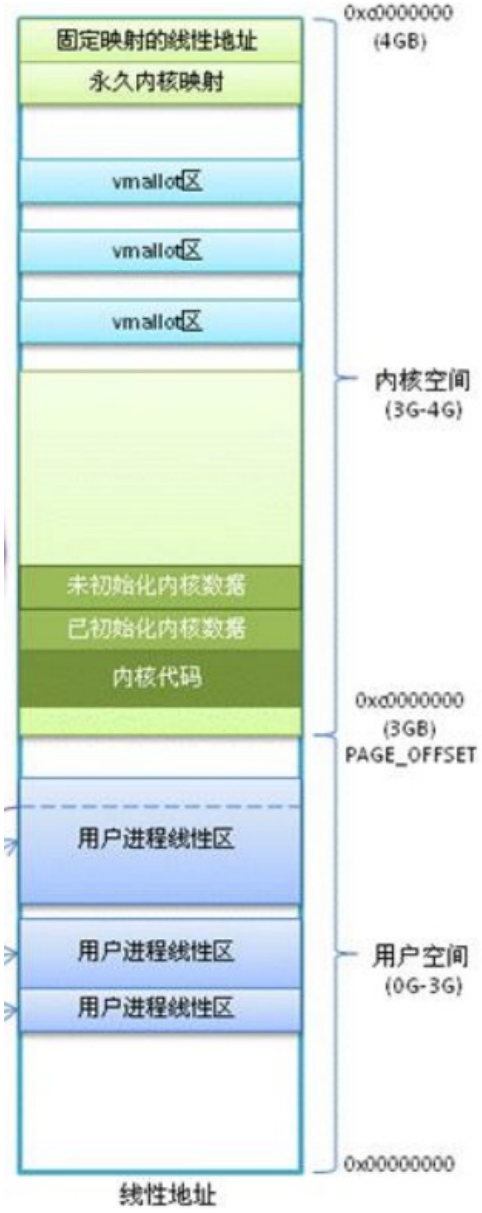
（2）、物理内存的分配。

1.1、内存管理模型

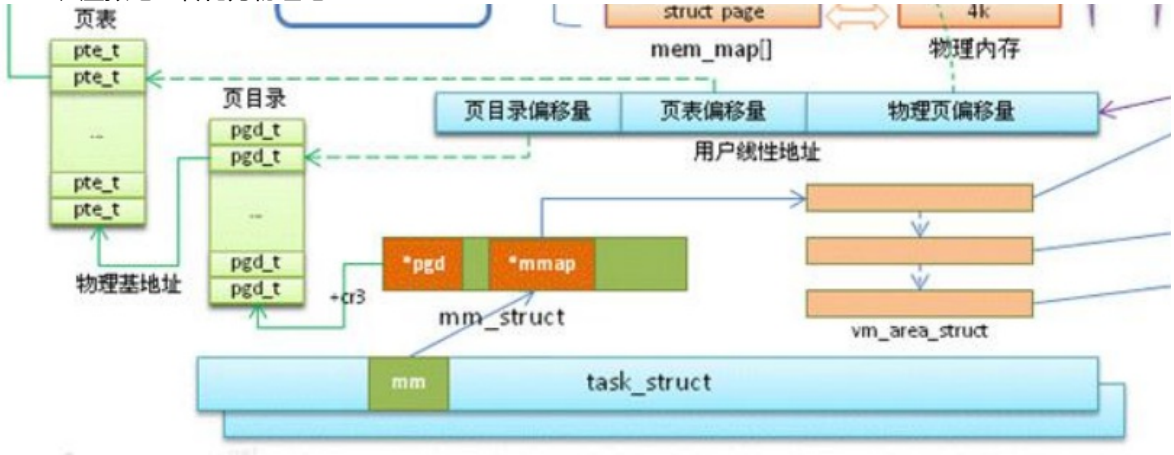


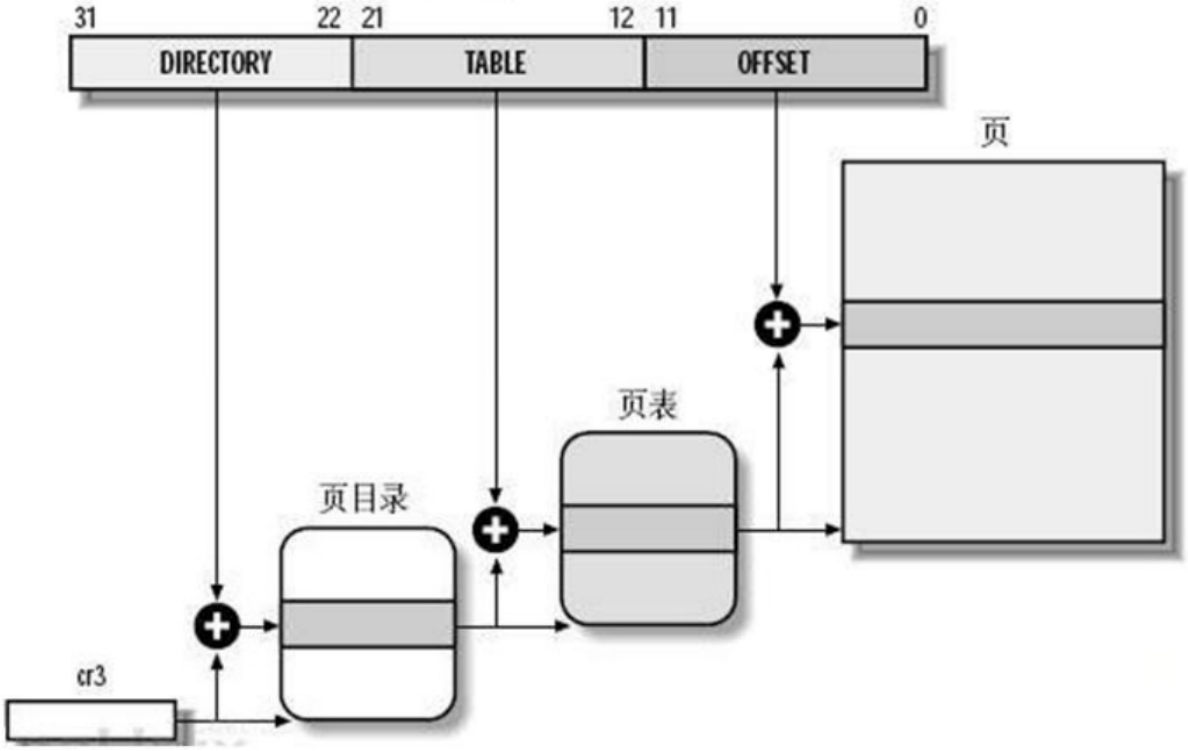
1.2、地址映射管理

1.2.1、虚拟地址空间分布

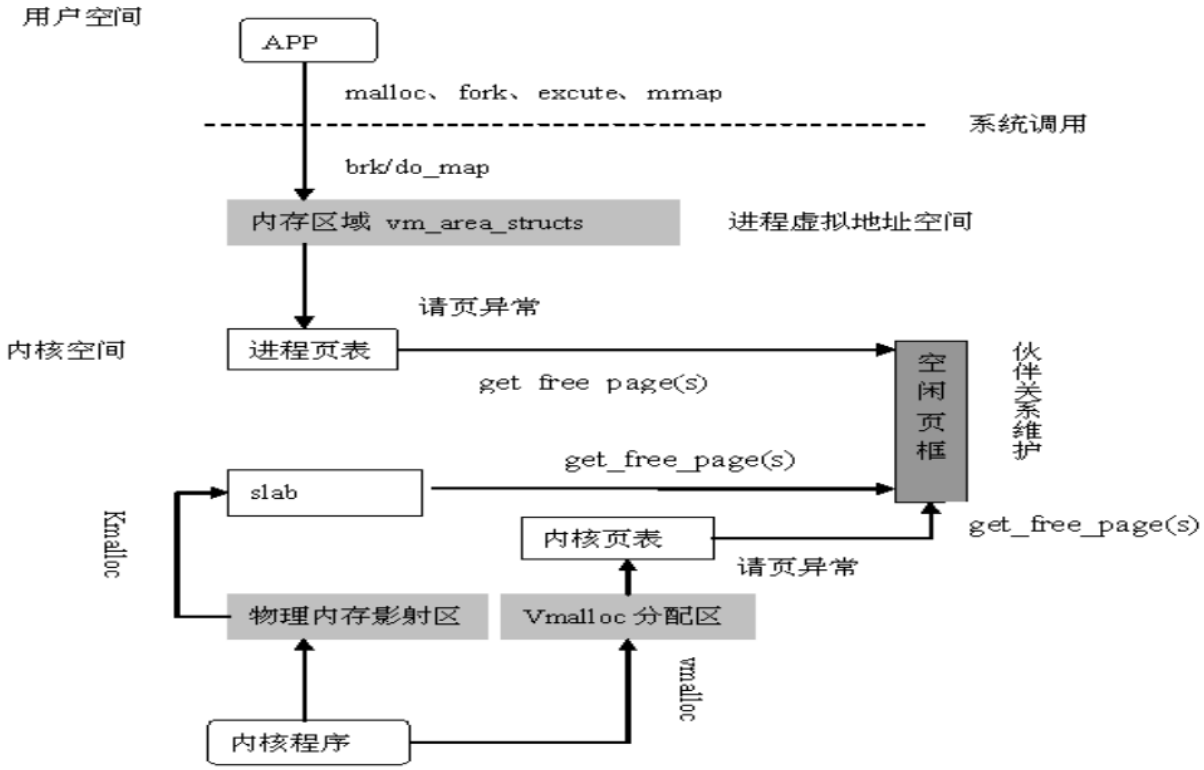


1.2.2、虚拟地址转化为物理地址





1.3、物理内存分配管理



只有当用户需要访问时，才会分配实际的物理内存。

二、Linux进程管理子系统

2.1、Linux进程要素

2.1.1、程序与进程

程序

存放在磁盘上的一系列代码和数据的可执行映像，是一个静止的实体

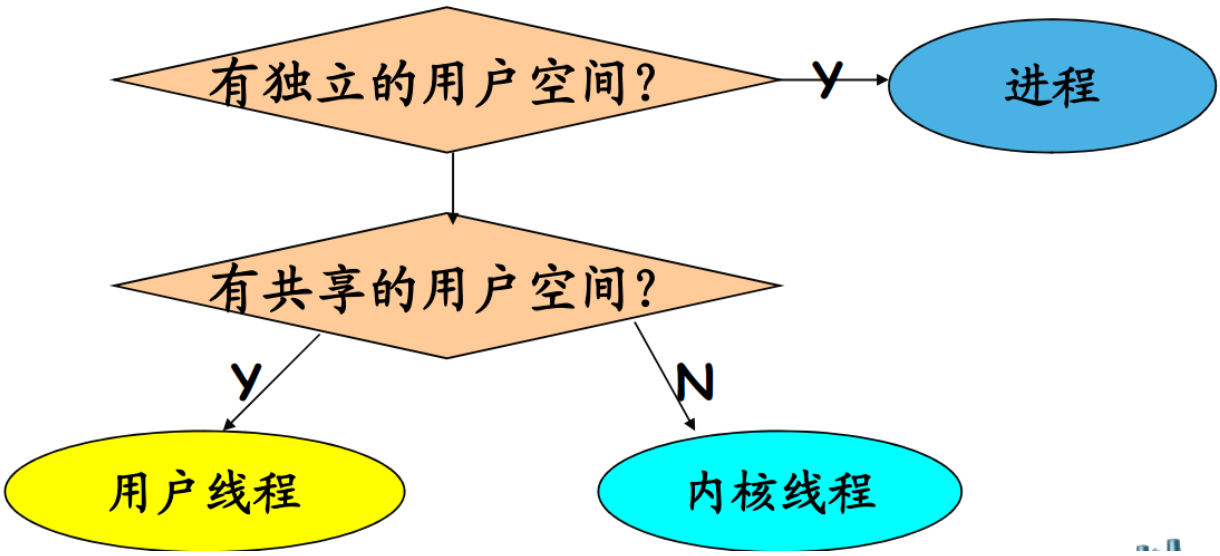
进程

是一个执行中的程序,它是动态的实体

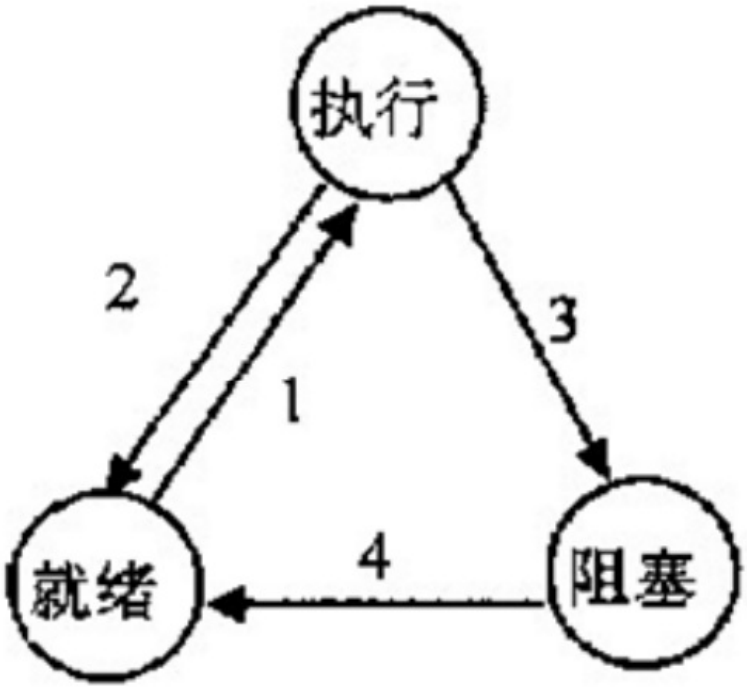
2.1.2、进程四要素

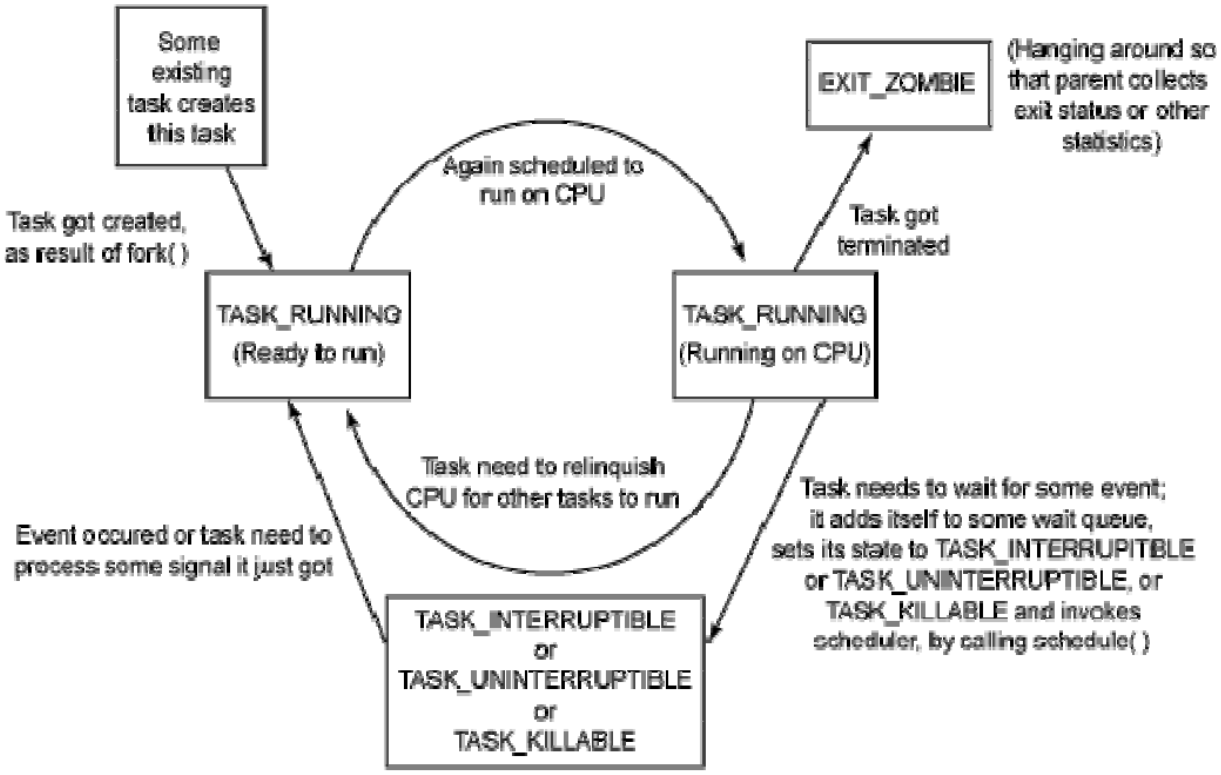
1. 有一段程序供其执行。这段程序不一定是某个进程所专有，可以与其他进程共用。

2. 有进程专用的内核空间堆栈。
3. 在内核中有一个task_struct数据结构，即通 常所说的“进程控制块”。有了这个数据结构，进程才能成为内核调度的一个基本单位接受内核的调度。
4. 有独立的用户空间。



2.1.3、Linux进程状态





比较重要的六个进程状态：

1. TASK_RUNNING
进程正在被CPU执行，或者已经准备就绪，随时可 以执行。当一个进程刚被创建时，就处于TASK_RUNNING状态。
2. TASK_INTERRUPTIBLE
处于等待中的进程，待等待条件为真时被唤醒，也 可以被信号或者中断唤醒。
3. TASK_UNINTERRUPTIBLE
处于等待中的进程，待资源有效时唤醒，但不可 以由其它进程通过信号(signal)或中断唤醒。
4. TASK_KILLABLE
Linux2.6.25新引入的进程睡眠状态，原理类似 于TASK_UNINTERRUPTIBLE，但是可以被致命信号(SIGKILL)唤醒。
5. TASK_TRACED
正处于被调试状态的进程。
6. TASK_DEAD
进程退出时(调用do_exit)，所处的状态。

2.1.4、进程描述结构

在Linux内核代码中，线程、进程都使用结构 task_struct(sched.h)来表示，它包含了大量描 述进程/线程的信息，其中比较重要的有：

pid_t pid; //进程号
vlong state; //进程状态
vint prio; //进程优先级

2.2、Linux进程调度

2.2.1、调度策略

SCHED_NORMAL(SCHED_OTHER):普通的分 时进程
SCHED_FIFO :先入先出的实时进程
SCHED_RR：时间片轮转的实时进程
SCHED_BATCH:批处理进程
SCHED_IDLE: 只在系统空闲时才能够被调度执 行的进程

2.2.2、调度时机

主动式

在内核中直接调用schedule()。当进程需要等待资 源等而暂时停止运行时，会把自己的状态置于挂起（睡眠），并主动请求调度，让出CPU。

范例:

```
1. current->state = TASK_INTERRUPTIBLE;
2. schedule();
```

被动式

被动式调度又名：抢占式调度。分为：用户态抢 占(Linux2.4、Linux2.6)和内核态抢占(Linux2.6)。

调度时机-用户态抢占

用户抢占发生在：

从系统调用返回用户空间。

从中断处理程序返回用户空间。 内核即将返回用户空间的时候，如果need_resched标志被设置，会导致schedule()被调用，即发生用户抢占。

当某个进程耗尽它的时间片时，会设置need_resched标志

当一个优先级更高的进程进入可执行状态的时候，也会设置 need_resched标志。

调度时机-内核态抢占

用户态抢占缺陷

进程/线程一旦运行到内核态，就可以一直执行，直 到它主动放弃或时间片耗尽为止。这样会导致一些非常紧急的进程或线程将长时间得不到运行，降低整个系统 的实时性。

改进方式

允许系统在内核态也支持抢占，更高优先级的进程/线 程可以抢占正在内核态运行的低优先级进程/线程。

内核抢占可能发生在：

中断处理程序完成，返回内核空间之前。

当内核代码再一次具有可抢占性的时候， 如解锁及使能软中断等。

在支持内核抢占的系统中，某些特例下是不允许抢占的：

内核正在运行中断处理。

内核正在进行中断上下文的Bottom Half(中断的底半部)处理。硬件中断返回前会执行软中断，此时仍然处于中断上下文中。

进程正持有spinlock自旋锁、writelock/readlock读写锁等， 当持有这些锁时，不应该被抢占，否则由于抢占将可能导致其他进程长期得不到锁，而让系统处于死锁状态。

内核正在执行调度程序Scheduler。抢占的原因就是为了进行新的调度，没有理由将调度程序抢占掉再运行调度程序。

调度时机-抢占计数

为保证Linux内核在以上情况下不会被抢占，抢占式 内核使用了一个变量preempt_count，称为内核抢占计数。这一变量被设置在进程的thread_info结构中。每当内核要进入以上几种状态时，变量preempt_count就加1，指示内核不允许抢占。每当内核从以上几种状态退出时，变量preempt_count就减1，同时进行可抢占的判断与调度。

2.2.3、调度步骤

Schedule函数工作流程如下：

1). 清理当前运行中的进程；

2). 选择下一个要运行的进程；

3). 设置新进程的运行环境；

4). 进程上下文切换 。