## 第一章 绪论

### linux特点

单块结构内核：区别于微内核

模块支持：学习微内核的优点

多线程

抢占式

多处理器

文件系统

### linux版本

四个数字：x1.x2.x3.x4

x1：主版本，自1996年一直是2

x2：次版本，奇数为开发版本，偶数为稳定版问

x3：发行版本

x4：添加补丁版本

### 内核概述

用户态和内核态的功能是为了防止用户程序直接操作硬件。

可重入内核：若干进程可同时在内核态下运行，实现手段是（可重入函数）或者（非重入函数+锁），可重入函数只修改局部变量，非重入函数使用锁机制保证临界区中只有一个进程。

### 文件系统概述

软链接：ln -s P1 P2

P2指向路径名P1，文件系统创建一个名为P2的软链接类型的新项，这个文件包含路径名P1。

## 第二章 内存寻址

### 三种内存地址

逻辑地址：段基址+偏移量

线性地址：虚拟地址，32位无符号整数

物理地址：用于内存芯片级内存单元寻址

虚拟地址

物理地址地址

逻辑地址

分段单元

分页单元

### 硬件中的分段

#### 段寄存器

cs：代码段寄存器

ds：数据段寄存器

ss：栈寄存器

还有3个通用段寄存器：es，fs，gs

#### 段描述符（8字节）

代码段描述符：LDT或者GDT，S=1(非系统段)

数据段描述符：LDT或者GDT，S=1

任务状态段描述符：GDT，S=0(系统段)

#### 段选择符

|  |  |  |
| --- | --- | --- |
| Index | TI | RPL |

index：13字节，GDT/LDT中的索引

TI：table indicator，表明GDT（0）还是LDT（1）

RPL：特权级

#### 分段单元

80x86处理器提供一种不可编程寄存器8字节，用于保存段描述符。

不可编程寄存器

|  |
| --- |
| 段描述符 |

base，limit

段寄存器

|  |
| --- |
| 段描述符 |

段

|  |
| --- |
|  |

具体过程描述：

1. 将段描述符填入段寄存器，检测TI，判断GDT/LDT；查看index，从GDT/LDT中填充段描述符到不可编程寄存器；
2. 得到段基址Base，Base+offset位线性地址，会判断是否超出limit限制。

除非段寄存器改变，否则不执行步骤1。

#### 分段和分页

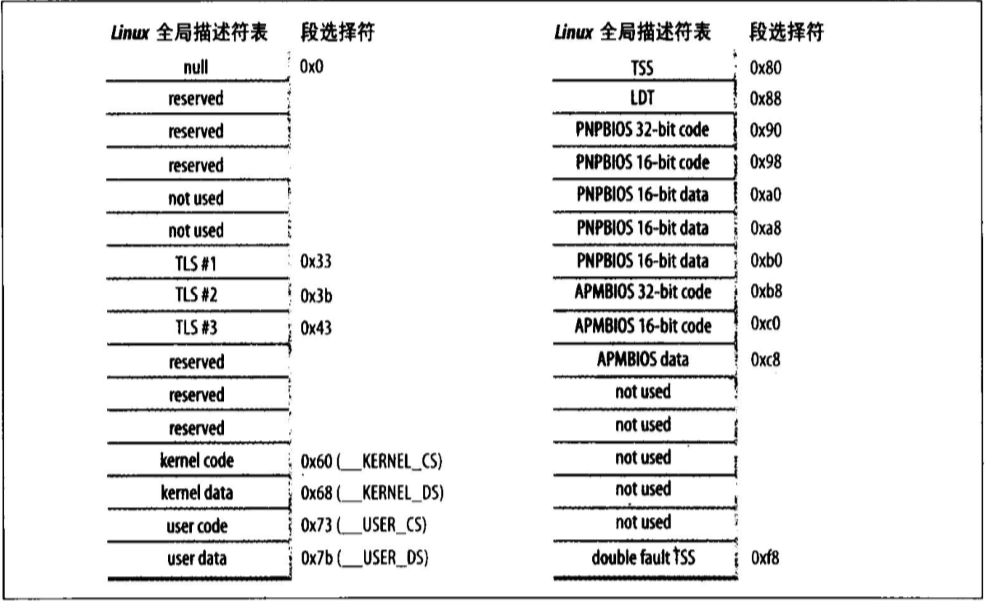
分段给每一个进程分配不同的线性地址空间；分页将同一线性地址映射到不同的物理地址。

### linux中的分段

linux中实际情况：base=0x00000000，limit=0xfffff，所有进程使用同一个base和limit，即实际中逻辑地址和线性地址是相同的。

#### GDT

1. 每个cpu对应一个GDT，所有GDT存放在cpu\_gdt\_tabl数组中，所有gdt地址和大小（用于初始化gdtr）存放在cpu\_gdt\_descr数组中。符号文件位于/arch/i386/kernel/head.S中。
2. GDT中包含18个段描述符和14个空、未使用或者保留的项。插入未使用的项为了使经常一起访问的描述符处于同一个32字节的硬件高速缓存中。
3. 线程局部存储（thread local storage） 3个
4. 用户态/内核态 数据段/代码段 4个
5. tss 1个
6. 默认ldt 1个
7. PnP（即插即用） 5个
8. APM（高级电源管理） 3个
9. 双重错误tss 1个



#### LDT

大多数用户态的linux程序不使用局部描述符。内核定义了一个缺省的ldt项，大多数进程共享。缺省的ldt存储在default\_ldt中。

### 硬件中的分页

1. 页框和页：页是数据块，可以存放在页框或者磁盘中。
2. cr0中的PG标志位置位启用分页；cr2存储发生缺页异常时的线性地址；cr3存储页目录表的物理地址；cr4中PGE标志位置位启用全局页，防止常用页从TLB(transform lookaside buffer)中置换出去。
3. 页目录表中的项和页表中的项结构一样：

|  |  |
| --- | --- |
| 高20位地址 | 12位标志位 |

1. Present：表示页是否在内存中
2. Accessed：该页是否被访问过，分页单元置位，操作系统重置
3. Dirty：只存在于页表项，操作系统重置
4. USER/SUPERVISOR：特权级
5. PCD/PWT：控制高速缓存处理页或页表的方式
6. Page Size：应用于页目录项，置1表示使用2M或4M页
7. Global：只用于页表项，防止常用页从TLB中被置换出去，当cr4中PGE位置位时起作用
8. 扩展分页
9. 允许页大小位2M或者4M，省略页表
10. 设置页目录项中Page Size标志位，只有高10位表物理地址
11. 通过设置cr4中PSE标志位，使扩展分页和常规分页共存
12. 硬件保护方案

|  |  |  |
| --- | --- | --- |
|  | 特权级 | 存取权限 |
| 分段 | 0，1，2，3 | R/W/E |
| 分页 | <3，3 | R/W |

1. PAE（物理地址扩展）分页机制
2. intel使用36根地址线，可寻址64G物理地址，但是线性地址还是32位（32位操作系统），所以需要PAE分页机制。
3. 将RAM分为224页框（以前是220），所以页目录项和页表项中物理地址需要高24位，加上12位标志位24+12=36，所以只能使用64位表示一项，那么4KB的页中页目录表和页表由原来的1024项变为512表项。
4. PDPT（页目录指针表），存储4个指针，分别指向4个页目录表，cr3存储27位（4个指针32位，首地址必然是32的倍数，25，5+27=32）的PDPT首地址。以前使只有一个页目录表，cr3直接存储页目录表的物理地址。
5. 32位线性地址解释：

|  |  |
| --- | --- |
| 31-30 | PDPT索引 |
| 29-21 | 页目录表索引 |
| 20-12 | 页表索引 |
| 11-0 | 页内偏移 |

1. 硬件高速缓存
2. 为了缩小cpu和ram之间的速度不匹配
3. 存在于主存和分页单元之间
4. 分为通写和回写
5. cr0中CD位启用或者禁止高速缓存，NW位选择通写还是回写
6. linux中PCD和PWT位清除，表示所有页框启用高速缓存，并使用回写策略
7. TLB

缓存线性地址和物理地址

### linux中的分页

高4G线性地址留给内核使用，其中128M线性地址空间实现非连续内存分配和固定映射的线性地址。

## 第三章 进程

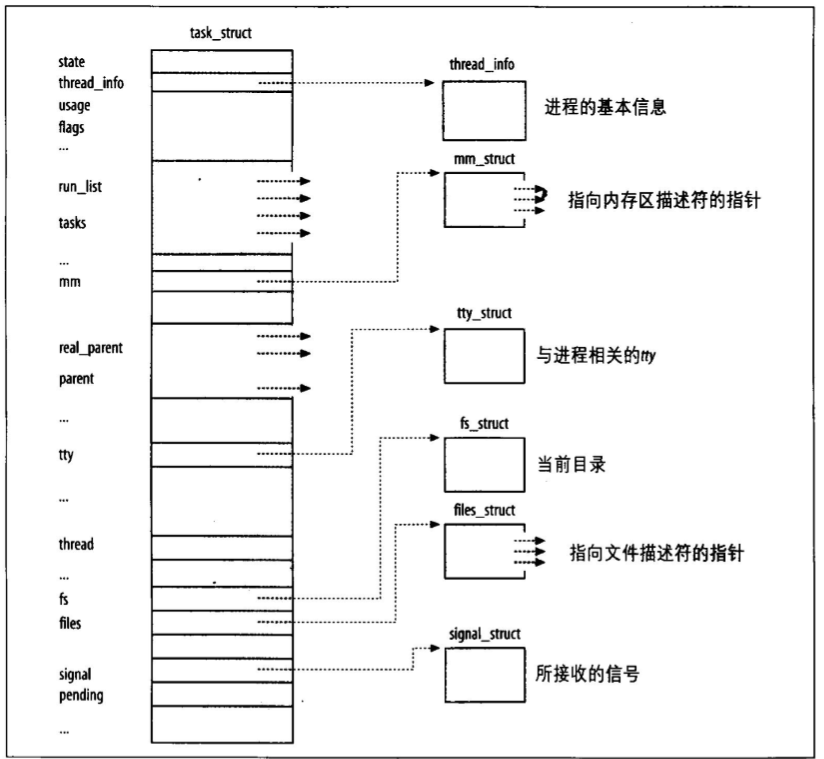
### 进程、轻量级进程和线程

如果内核不支持多线程应用，所有的执行流都是在用户态下进程，从内核角度来看，它们只是一个进程。例如，一个象棋程序，两个线程，分别时等待用户输入和思考下一步落子，当线程1等待用户输入时，不能阻塞进程，否则线程2也被阻塞。

现代unix系统支持多进程应用，linux使用轻量级进程提供对多线程应用程序的支持，两个轻量级进程共享一些资源，当然必须处理同不问题。

### 进程描述符

task\_struct：



#### 进程状态：

可运行状态（TASK\_RUNNING）

可中断等待状态（TASK\_INTERRUPTIBLE）

不可中断等待状态（TASK\_UNINTERRUPTIBLE）

暂停状态（TASK\_STOPPED）

跟踪状态（TASK\_TRACED）

还有两种状态还可以作为exit\_state：

僵尸状态（EXIT\_ZOMBIE）

死亡状态（EXIT\_DEAD）

#### 进程标识

PID循环使用，使用pidmap-array用作位图。

线程组中所有轻量级进程使用同一个PID。一个线程组中所有线程使用和该组中领头线程的pid，它被存入进程描述符的tgip中，getpid（）返回tgip而不是pid。

#### 进程描述符处理

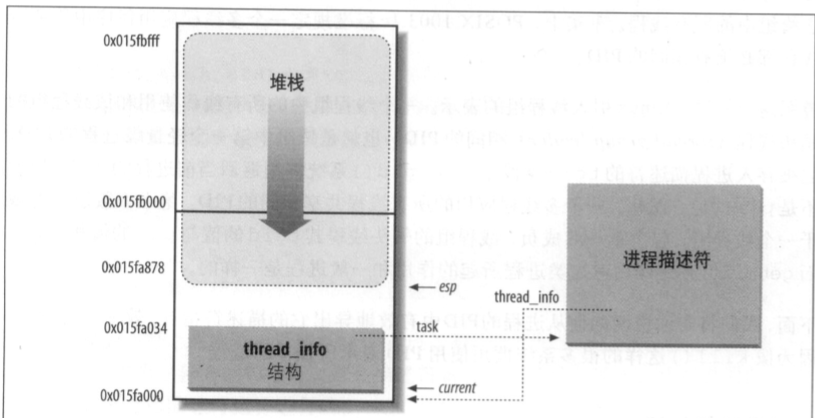
对于每一个进程，linux把进程相关的两个数据结构紧凑的存放在一个单独为进程分配的存储区域：

union thread\_info{

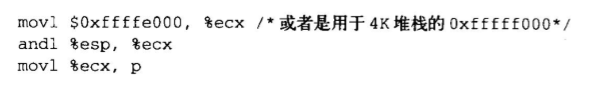
struct thread\_info thread\_info;

unsigned long stack[2048];/\*对于4K的栈数组下标是1024\*/

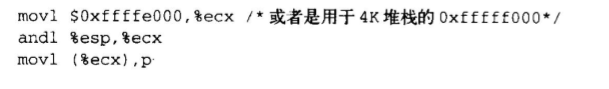
}



标识当前进程，内核很容易从esp寄存器的值获得当前在cpu上正在运行的进程的thread\_info结构的地址。事实上，如果thread\_union的结构长度是8K（213），则内核屏蔽掉低13位即就是thread\_info的地址。



当要获取进程描述符地址时，内核调用current宏，该宏 本质等价于current\_thread\_info()->task，产生如下汇编语言指令：



因为task字段在thread\_info结构中偏移量位0.