



平台：论坛 博客 Club168 精华 文库 自测 访谈录| 频道：操作系统 开发 数据库 存储 服务器 网络 IT新闻 Linux 下载 Power用户组 搜索


·[买域名送域名 海外主机免备案](#) ·[CU自测第2期CU币兑换活动开始啦](#) ·[数据防泄密有奖大讨论](#)

论坛 程序设计 内核源码 我理解的逻辑地址、线性地址、物理地址和虚拟地址(补充完 ...

发表主题

版块跳转 最近访问板块 1 2 3 4 5 6 7 8 9 10 ... 22 下一页

查看: 134191 | 回复: 213



富足长乐  
帖子 2356  
主题 424  
精华 36  
可用积分 5927  
专家积分 0  
在线时间 554 小时  
注册时间 2003-08-12  
最后登录 2012-07-02  
串门 好友  
博客 消息  
论坛徽章: 0

### 我理解的逻辑地址、线性地址、物理地址和虚拟地址(补充完整了)

发表于 2008-01-15 16:32:15 | 只看该作者 | 倒序浏览

[报告] [收藏(0)] 1楼 电梯直达

要过年了，发个年终总结贴，只是个人理解，不包正确哈。

本贴涉及的硬件平台是X86，如果是其它平台，嘻嘻，不保证能一一对号入座，但是举一反三，我想是完全可行的。

#### 一、概念

##### 物理地址(physical address)

用于内存芯片级的单元寻址，与处理器和CPU连接的地址总线相对应。

——这个概念应该是这几个概念中最好理解的一个，但是值得一提的，虽然可以直接把物理地址理解成插在机器上那根内存本身，把内存看成一个从0字节一直到最大空量逐字节的编号的大数组，然后把这个数组叫做物理地址，但是事实上，这只是一个硬件提供给软件的抽象，内存的寻址方式并不是这样。所以，说它是“与地址总线相对应”，是更贴切一些，不过抛开对物理内存寻址方式的考虑，直接把物理地址与物理的内存一一对应，也是可以接受的。也许错误的理解更利于形而上的抽象。

##### 虚拟内存(virtual memory)

这是对整个内存（不要与机器上插那条对上号）的抽象描述。它是相对于物理内存来讲的，可以直接理解成“不直实的”，“假的”内存，例如，一个0x08000000内存地址，它并不就对就物理地址上那个大数组中0x08000000 - 1那个地址元素；之所以是这样，是因为现代操作系统都提供了一种内存管理的抽象，即虚拟内存（virtual memory）。进程使用虚拟内存中的地址，由操作系统协助相关硬件，把它“转换”成真正的物理地址。这个“转换”，是所有问题讨论的关键。

有了这样的抽象，一个程序，就可以使用比真实物理地址大得多的地址空间。（拆东墙，补西墙，银行也是这样子做的），甚至多个进程可以使用相同的地址。不奇怪，因为转换后的物理地址并非相同的。

——可以把连接后的程序反编译看一下，发现连接器已经为程序分配了一个地址，例如，要调用某个函数A，代码不是call A，而是call 0x08111111，也就是说，函数A的地址已经被定下来了。没有这样的“转换”，没有虚拟地址的概念，这样做是根本行不通的。打住了，这个问题再说下去，就收不住了。

##### 逻辑地址(logical address)

Intel为了兼容，将远古时代的段式内存管理方式保留了下来。逻辑地址指的是机器语言指令中，用来指定一个操作数或者是一条指令的地址。以上例，我们说的连接器为A分配的0x08111111这个地址就是逻辑地址。

——不过不好意思，这样说，好像又违背了Intel中段式管理中，对逻辑地址要求，“一个逻辑地址，是由一个段标识符加上一个指定段内相对地址的偏移量，表示为 [段标识符: 段内偏移量]，也就是说，上例中那个0x08111111，应该表示为[A的代码段标识符: 0x08111111]，这样，才完整一些”

##### 线性地址(linear address)或也叫虚拟地址(virtual address)

跟逻辑地址类似，它也是一个不真实的地址，如果逻辑地址是对应的硬件平台段式管理转换前地址的话，那么线性地址则对应了硬件页式内存的转换前地址。

-----

CPU将一个虚拟内存空间中的地址转换为物理地址，需要进行两步：首先将给定一个逻辑地址（其实是段内偏移量，这个一定要理解！！），CPU要利用其段式内存管理单元，先将为个逻辑地址转换成一个线性地址，再利用其页式内存管理单元，转换为最终物理地址。

这样做两次转换，的确是非常麻烦而且没有必要的，因为直接可以把线性地址抽象给进程。之所以这样冗余，Intel完全是为了兼容而已。

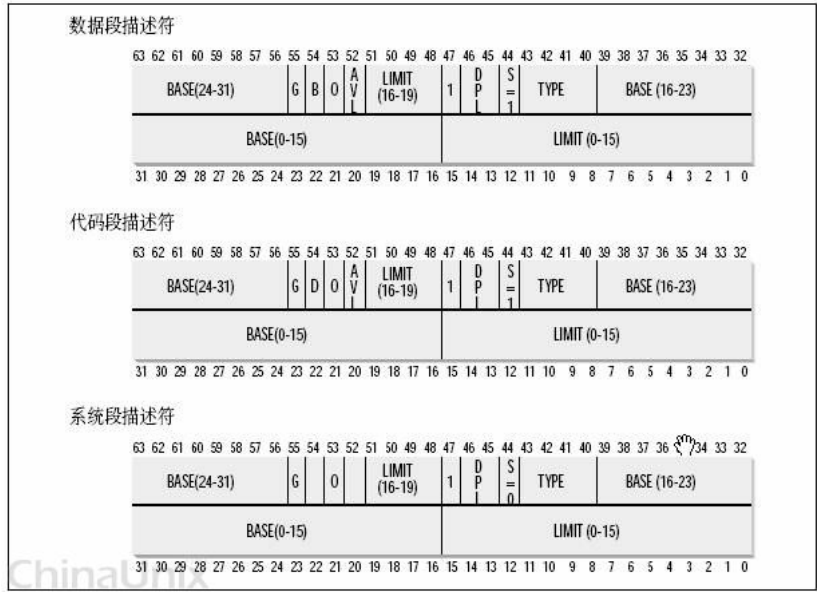
## 2、CPU段式内存管理，逻辑地址如何转换为线性地址

一个逻辑地址由两部份组成，段标识符: 段内偏移量。段标识符是由一个16位长的字段组成，称为段选择符。其中前13位是一个索引号。后面3位包含一些硬件细节，如图：

最后两位涉及权限检查，本贴中不包含。

索引号，或者直接理解成数组下标——那它总要对应一个数组吧，它又是什么东东的索引呢？这个东东就是“段描述符(segment descriptor)”，呵呵，段描述符具体地址描述了一个段（对于“段”这个字眼的理解，我是把它想像成，拿了一把刀，把虚拟内存，砍成若干的截——一段）。这样，很多个段描述符，就组了一个数组，叫“段描述符表”，这样，可以通过段标识符的前13位，直接在段描述符表中找到一个具体的段描述符，这个描述符就描述了一个段，我刚才对段的抽象不太准确，因为看看描述符里面究竟有什么东东——也就是它究

竟是如何描述的，就理解段究竟有什么东东了，每一个段描述符由8个字节组成，如下图：

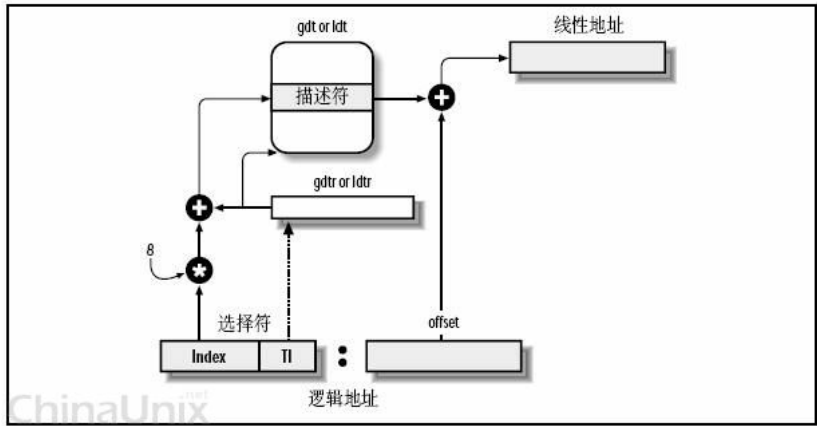


这些东东很复杂，虽然可以利用一个数据结构来定义它，不过，我这里只关心一样，就是Base字段，它描述了一个段的开始位置的线性地址。

Intel设计的本意是，一些全局的段描述符，就放在“全局段描述符表(GDT)”中，一些局部的，例如每个进程自己的，就放在所谓的“局部段描述符表(LDT)”中。那究竟什么时候该用GDT，什么时候该用LDT呢？这是由段选择符中的T1字段表示的，=0，表示用GDT，=1表示用LDT。

GDT在内存中的地址和大小存放在CPU的gdtr控制寄存器中，而LDT则在ldtr寄存器中。

好多概念，像绕口令一样。这张图看起来要直观些：



- 首先，给定一个完整的逻辑地址[段选择符：段内偏移地址]，
- 1、看段选择符的T1=0还是1，知道当前要转换是GDT中的段，还是LDT中的段，再根据相应寄存器，得到其地址和大小。我们就有了一个数组了。
  - 2、拿出段选择符中前13位，可以在这个数组中，查找到对应的段描述符，这样，它了Base，即基地址就知道了。
  - 3、把Base + offset，就是要转换的线性地址了。

还是挺简单的，对于软件来讲，原则上就需要把硬件转换所需的信息准备好，就可以让硬件来完成这个转换了。OK，来看看Linux怎么做的。

### 3、Linux的段式管理

Intel要求两次转换，这样虽说是兼容了，但是却是很冗余，呵呵，没办法，硬件要求这样做了，软件就只能照办，怎么着也得形式主义一样。

另一方面，其它某些硬件平台，没有二次转换的概念，Linux也需要提供一个高层抽象，来提供一个统一的界面。所以，Linux的段式管理，事实上只是“哄骗”了一下硬件而已。

按照Intel的本意，全局的用GDT，每个进程自己的用LDT——不过Linux则对所有的进程都使用了相同的段来对指令和数据寻址。即用户数据段，用户代码段，对应的，内核中的是内核数据段和内核代码段。这样做没有什么奇怪的，本来就是走形式嘛，像我们写年终总结一样。

include/asm-i386/segment.h

```
01. #define GDT_ENTRY_DEFAULT_USER_CS    14
02. #define __USER_CS (GDT_ENTRY_DEFAULT_USER_CS * 8 + 3)
03.
04. #define GDT_ENTRY_DEFAULT_USER_DS    15
```

```
05.  #define __USER_DS (GDT_ENTRY_DEFAULT_USER_DS * 8 + 3)
06.
07.  #define GDT_ENTRY_KERNEL_BASE      12
08.
09.  #define GDT_ENTRY_KERNEL_CS          (GDT_ENTRY_KERNEL_BASE + 0)
10.  #define __KERNEL_CS (GDT_ENTRY_KERNEL_CS * 8)
11.
12.  #define GDT_ENTRY_KERNEL_DS          (GDT_ENTRY_KERNEL_BASE + 1)
13.  #define __KERNEL_DS (GDT_ENTRY_KERNEL_DS * 8)
```

复制代码

把其中的宏替换成数值，则为：

```
01.  #define __USER_CS 115      [00000000 1110  0  11]
02.  #define __USER_DS 123      [00000000 1111  0  11]
03.  #define __KERNEL_CS 96     [00000000 1100  0  00]
04.  #define __KERNEL_DS 104    [00000000 1101  0  00]
```

复制代码

方括号后是这四个段选择符的**16**位二进制表示，它们的索引号和**T1**字段值也可以算出来了

```
01.  __USER_CS          index= 14  T1=0
02.  __USER_DS          index= 15  T1=0
03.  __KERNEL_CS        index= 12  T1=0
04.  __KERNEL_DS        index= 13  T1=0
```

复制代码

T1均为**0**，则表示都使用了GDT，再来看初始化GDT的内容中相应的**12-15**项(arch/i386/head.S)：

```
01.          .quad 0x00cf9a000000ffff      /* 0x60 kernel 4GB code at 0x00000000 */
02.          .quad 0x00cf92000000ffff      /* 0x68 kernel 4GB data at 0x00000000 */
03.          .quad 0x00cffa000000ffff      /* 0x73 user 4GB code at 0x00000000 */
04.          .quad 0x00cff2000000ffff      /* 0x7b user 4GB data at 0x00000000 */
```

复制代码

按照前面段描述符表中的描述，可以把它们展开，发现其**16-31**位全为**0**，即四个段的基地址全为**0**。

这样，给定一个段内偏移地址，按照前面转换公式，**0 + 段内偏移**，转换为线性地址，可以得出重要的结论，“在Linux下，逻辑地址与线性地址总是一致（是一致，不是有些人说的相同）的，即逻辑地址的偏移量字段的值与线性地址的值总是相同的。！！！”

忽略了太多的细节，例如段的权限检查。呵呵。

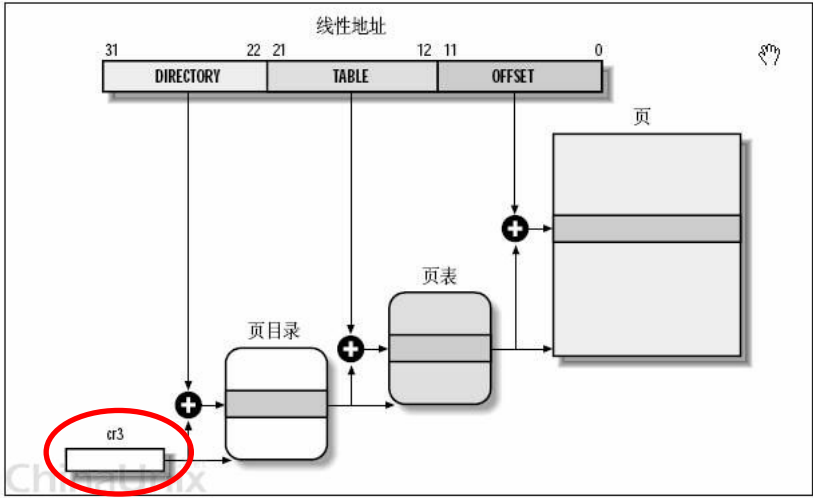
Linux中，绝大部份进程并不例用LDT，除非使用Wine， 仿真Windows程序的时候。

## 4.CPU的页式内存管理

CPU的页式内存管理单元，负责把一个线性地址，最终翻译为一个物理地址。从管理和效率的角度出发，线性地址被分为以固定长度为单位的组，称为页(page)，例如一个32位的机器，线性地址最大可为**4G**，可以用**4KB**为一个页来划分，这页，整个线性地址就被划分为一个total\_page[2^20]的大数组，共有2的20个次方个页。这个大数组我们称之为页目录。目录中的每一个目录项，就是一个地址——对应的页的地址。

另一类“页”，我们称之为物理页，或者是页框、页帧的。是分页单元把所有的物理内存也划分为固定长度的管理单位，它的长度一般与内存页是一一对应的。

这里注意到，这个total\_page数组有2^20个成员，每个成员是一个地址（32位机，一个地址也就是4字节），那么要单单要表示这么一个数组，就要占去4MB的内存空间。为了节省空间，引入了一个二级管理模式的机器来组织分页单元。文字描述太累，看图直观一些：



- 如上图，
- 1、分页单元中，页目录是唯一的，它的地址放在CPU的cr3寄存器中，是进行地址转换的开始点。万里长征从此长始了。
  - 2、每一个活动的进程，因为都有其独立的对应的虚拟内存（页目录也是唯一的），那么它也对了一个独立的页目录地址。——运行一个进程，需要将它页目录地址放到cr3寄存器中，将别的保存下来。
  - 3、每一个32位的线性地址被划分为三部份，页目录索引(10位)：页表索引(10位)：偏移(12位)
- 依据以下步骤进行转换：
- 1、从cr3中取出进程的页目录地址（操作系统负责在调度进程的时候，把这个地址装入对应寄存器）；
  - 2、根据线性地址前十位，在数组中，找到对应的索引项，因为引入了二级管理模式，页目录中的项，不再是页的地址，而是一个页表的地址。（又引入了一个数组），页的地址被放到页表中去了。
  - 3、根据线性地址的中间十位，在页表（也是数组）中找到页的起始地址；
  - 4、将页的起始地址与线性地址中最后12位相加，得到最终我们想要的葫芦；

这个转换过程，应该说还是非常简单的。全部由硬件完成，虽然多了一道手续，但是节约了大量的内存，还是值得的。那么再简单地验证一下：

- 1、这样的二级模式是否仍能够表示4G的地址：

页目录共有：2^10项，也就是说有这么多个页表  
每个目录对应了：2^10页；  
每个页中可寻址：2^12个字节。  
还是2^32 = 4GB

2、这样的二级模式是否真的节约了空间；  
也就是算一下页目录项和页表项共占空间 (2^10 \* 4 + 2^10 \* 4) = 8KB。哎，.....怎么说呢！！  
红色错误，标注一下，后文贴中有此讨论。。。。。

按<深入理解计算机系统>中的解释,二级模式空间的节约是从两个方面实现的：

- A、如果一级页表中的一个页表条目为空，那么那所指的二级页表就根本不会存在。这表现出一种巨大的潜在节约，因为对于一个典型的程序，4GB虚拟地址空间的大部份都会是未分配的；
- B、只有一级页表才需要总是在主存中。虚拟存储器系统可以在需要时创建，并页面调入或调出二级页表，这就减少了主存的压力。只有最经常使用的二级页表才需要缓存在主存中。——不过Linux并没有完全享受这种福利，它的页表目录和与已分配页面相关的页表都是常驻内存的。

值得一提的是，虽然页目录和页表中的项，都是4个字节，32位，但是它们都只用高20位，低12位屏蔽为0——把页表的低12屏蔽为0，是很好理解的，因为这样，它刚好和一个页面大小对应起来，大家都成整数增加。计算起来就方便多了。但是，为什么同时也要把页目录低12位屏蔽掉呢？因为按同样的道理，只要屏蔽其低10位就可以了，不过我想，因为12>10，这样，可以让页目录和页表使用相同的数据结构，方便。

本贴只介绍一般性转换的原理，扩展分页、页的保护机制、PAE模式的分页这些麻烦点的东东就不啰嗦了.....可以参考其它专业书籍。

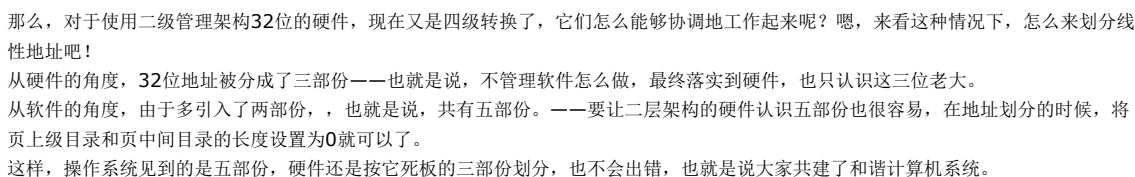
### 5.Linux的页式内存管理

原理上来讲，Linux只需要为每个进程分配好所需数据结构，放到内存中，然后在调度进程的时候，切换寄存器cr3，剩下的就交给硬件来完成了（呵呵，事实上要复杂得多，不过偶只分析最基本的流程）。

前面说了i386的二级页管理架构，不过有些CPU，还有三级，甚至四级架构，Linux为了在更高层次提供抽象，为每个CPU提供统一的界面。提供了一个四层页管理架构，来兼容这些二级、三级、四级管理架构的CPU。这四级分别为：

- 页全局目录PGD（对应刚才的页目录）
- 页上级目录PUD（新引进的）
- 页中间目录PMD（也就新引进的）
- 页表PT（对应刚才的页表）。

整个转换依据硬件转换原理，只是多了二次数组的索引罢了，如下图：



例如，一个逻辑地址已经被转换成了线性地址，0x08147258，换成二进制，也就是：

```
0000100000 0101000111 001001011000
```

内核对这个地址进行划分





```
PGD = 0000100000
PUD = 0
PMD = 0
PT = 0101000111
offset = 001001011000
```

从软件的角度上来讲，因为它的项只有一个，32位，刚好可以存放与PGD中长度一样的地址指针。那么所谓先到PUD，到到PMD中做映射转换，就变成了保持原值不变，一一转手就可以了。这样，就实现了“逻辑上指向一个PUD，再指向一个PDM，但在物理上是直接指向相应的PT的这个抽象，因为硬件根本不知道有PUD、PMD这个东西”。

页目录 = 0000100000  
 PT = 0101000111  
 offset = 001001011000

[ 本帖最后由 独孤九贱 于 2009-9-22 20:36 编辑 ]

招个人，有意站内联系。地点：重庆，熟**Linux C**开发，有应用协议分析、熟**NFA/DFA**方面经验优先。  
<http://bbs.chinaunix.net/thread-3749620-1-1.html>

<div></div> <div>家境小康</div> <div></div> <div><div>帖子656</div><div>主题33</div><div>精华0</div><div>可用积分1063</div><div>专家积分0</div><div>在线时间7 小时</div><div>注册时间2007-03-12</div><div>最后登录2010-11-29</div></div> <div><div>串门</div><div>好友</div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div>Thomas Yao</div> <div>Shanghai Linux User Group</div> <div>Twitter.com/ghosTM55</div> <div>http://www.ghosTunix.org</div> <div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越! 参与数据防泄密有奖大讨论!</div>
<div>netentsec</div> <div></div> <div>白手起家</div> <div><div>帖子18</div><div>主题10</div><div>精华1</div><div>可用积分24</div><div>专家积分0</div><div>在线时间1 小时</div><div>注册时间2007-04-25</div><div>最后登录2009-06-17</div></div> <div><div>串门</div><div>好友</div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div> 发表于 2008-01-15 16:48:12   只看该作者</div> <div>[报告] 3楼</div> <div>深入浅出, 讲的好, 期待下文😊</div> <div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越! 参与数据防泄密有奖大讨论!</div>
<div>sisi8408</div> <div></div> <div>丰衣足食</div> <div><div>☆</div><div>帖子691</div><div>主题39</div><div>精华0</div><div>可用积分569</div><div>专家积分0</div><div>在线时间7 小时</div><div>注册时间2006-12-22</div><div>最后登录2011-01-16</div></div> <div><div>串门</div><div>好友</div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div> 发表于 2008-01-15 20:07:41   只看该作者</div> <div>[报告] 4楼</div> <div>赞一个。。。。。。。。。</div> <div>东直门外大街 张字85号 丁字96号</div> <div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越! 参与数据防泄密有奖大讨论!</div>
<div>duanius</div> <div></div> <div>小富即安</div>	<div> 发表于 2008-01-15 21:14:06   只看该作者</div> <div>[报告] 5楼</div> <div>说的不错 比ulk第二章更直白 人性化</div> <div>[ 本帖最后由 duanius 于 2008-1-15 21:15 编辑 ]</div>

帖子

914

主题

102

精华

2

可用积分

2176

专家积分

0

在线时间

324 小时

注册时间

2006-10-01

最后登录

2012-06-29

串门

好友

博客

消息

论坛徽章：0

Latitude/Optiplex 老用户征集活动 | 如何使企业中的服务器、数据信息更为安全 | Nginx在工作中的应用讨论 | 立即穿越！参与数据防泄密有奖大讨论！

flw2

大富大贵

😄😁🌟

帖子

5391

主题

172

精华

2

可用积分

11919

专家积分

10

在线时间

319 小时

注册时间

2005-11-19

最后登录

2011-08-20

串门

好友

博客

消息

论坛徽章：0

Latitude/Optiplex 老用户征集活动 | 如何使企业中的服务器、数据信息更为安全 | Nginx在工作中的应用讨论 | 立即穿越！参与数据防泄密有奖大讨论！

achlice

二手艺术家

巨富豪门

😄😁

帖子

6102

主题

804

精华

5

可用积分

33198

专家积分

210

在线时间

785 小时

注册时间

2005-11-05

最后登录

2012-03-26

串门

好友

博客

消息

论坛徽章：0

Latitude/Optiplex 老用户征集活动 | 如何使企业中的服务器、数据信息更为安全 | Nginx在工作中的应用讨论 | 立即穿越！参与数据防泄密有奖大讨论！

qps104

稍有积蓄

⭐

帖子

162

主题

12

精华

0

可用积分

208

专家积分

5

发表于 2008-01-16 09:34:41 | 只看该作者

[报告] 6楼

支持一个，LZ超级高手

发表于 2008-01-16 10:14:07 | 只看该作者

[报告] 7楼

见了好帖子然后回复，是一种美德，比如像我，一直都 是这样做的～～

看了内核源代码 情景 分析，从 8086 讲到80286， 80386，终于明白，这是怎么回事儿了～～

[ 本帖最后由 achlice 于 2008-1-28 15:53 编辑 ]






河南linuxer QQ群：45884249

发表于 2008-01-16 11:16:38 | 只看该作者

[报告] 8楼

好文,加深了我对3种地址的理解



<div>在线时间 87 小时</div> <div>注册时间 2007-02-28</div> <div>最后登录 2011-12-02</div> <div><div>串门</div><div>好友</div></div> <div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越！参与数据防泄密有奖大讨论！</div>
<div>yj1804</div> <div></div> <div>白手起家</div> <div>帖子 171</div> <div>主题 47</div> <div>精华 0</div> <div>可用积分 154</div> <div>专家积分 0</div> <div>在线时间 203 小时</div> <div>注册时间 2003-11-28</div> <div>最后登录 2012-06-20</div> <div><div>串门</div><div>好友</div></div> <div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div><div> 发表于 2008-01-16 17:20:30   只看该作者</div><div>[报告] 9楼</div></div> <div>好文,赞一个</div> <div>PS.推荐大家看毛德操的"LINUX内核源代码情景分析"一书, 里面讲解也比较详细</div> <div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越！参与数据防泄密有奖大讨论！</div>
<div>duanius</div> <div></div> <div>小富即安</div> <div></div> <div>帖子 914</div> <div>主题 102</div> <div>精华 2</div> <div>可用积分 2176</div> <div>专家积分 0</div> <div>在线时间 324 小时</div> <div>注册时间 2006-10-01</div> <div>最后登录 2012-06-29</div> <div><div>串门</div><div>好友</div></div> <div><div>博客</div><div>消息</div></div> <div>论坛徽章: 0</div>	<div><div> 发表于 2008-01-16 18:51:10   只看该作者</div><div>[报告] 10楼</div></div> <div>好文好文 隐隐中透着一种和谐美</div> <div>Latitude/Optiplex 老用户征集活动   如何使企业中的服务器、数据信息更为安全   Nginx在工作中的应用讨论   立即穿越！参与数据防泄密有奖大讨论！</div>

热门内容推荐
IT168产品库推荐: 苹果 iPod touch4(8G) 摩托罗拉 XT615 三星 i9003 索尼爱立信 X8 华为 S8600 火花 现代博恩 XB-D04

<div>发表主题</div>	<div>返回列表</div>	<div>1</div>	<div>2</div>	<div>3</div>	<div>4</div>	<div>5</div>	<div>6</div>	<div>7</div>	<div>8</div>	<div>9</div>	<div>10</div>	<div>... 22</div>	<div>下一页</div>	
-----------------	-----------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	---------------	-------------------	----------------	--

论坛 程序设计 内核源码 我理解的逻辑地址、线性地址、物理地址和虚拟地址(补充完 ...

	<div>高级模式</div> <div>您需要登录后才可以回帖 登录   注册  用QQ帐号登录</div> <div>发表回复 <input type="checkbox"/> 回帖后跳转到最后一页</div>
--	--