目的：真正理解内核是如何运作的

1. 物理地址

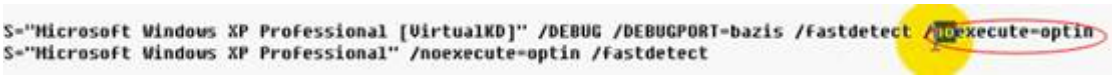
(1) Mov eax,dword ptr ds:[0x12345678]

其中0x12345678是有效地址

ds.Base + 0x12345678是线性地址

1.101012分页

1.C：\boot文件 如下图所示红圈内容 将no去掉，重启计算机



2.流程

线性地址 0x000AA8A0

0000 0000 00 010位 一级目录序号（PDT）

00 1010 1010 AA10位 二级目录序号（PTT）

8A0 8A0 12位 三级目录序号

(1)查找进程CR3寄存器（CR3指向一个物理页，一共4096字节）

(2)通过CR3所指地址为一级目录，使用一级目录序号检索二级目录

(3)使用二级目录序号检索三级目录

(4)使用三级目录序号检索到数据的位置

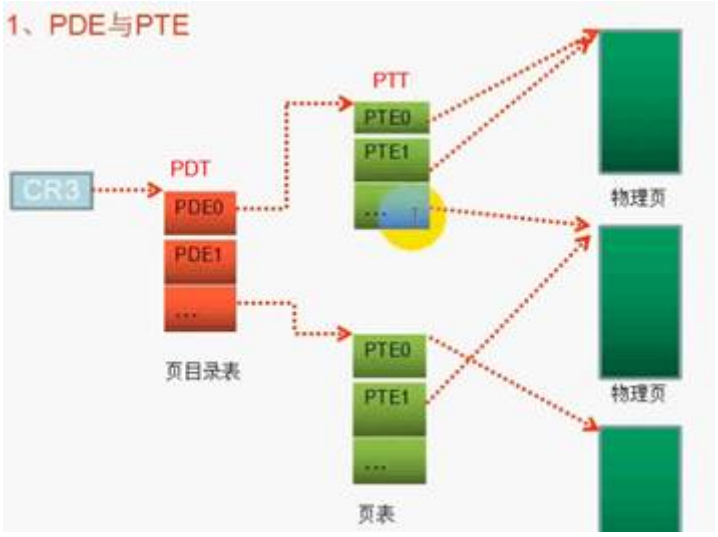
Windbg 指令集

!process 0 0查找进程Dr3 --->

获取DirBase ---> *[*[DirBase + 0 * 4] & 0xFFFFF000] + AA * 4] &0xFFFFF000 + 8A0

注 PDE 与 PTE后12位表示页属性,不计入地址

2.PDE与PTE



注 1.PTE可以没有物理页

2.多个PTE也可以指向同一个物理页

物理页挂靠：例进程0地址 修改PDE ， PTE属性 保证物理页可读，修改PTE，使其可以指向自己想要的物理页。

物理页的属性 = PDE属性 & PTE属性

R/W位 = 0 只读

R/W位 = 1 可读可写

（例）字符串常量不可修改

U/S位 = 0 特权用户

U/B位 = 1 普通用户

之前通过提权的方式，访问内存高2g的内存，现在我们可以尝试在三环修改页属性，从而访问高2g的内存

PS位：只对PDE有意义，PS == PageSize的意思，当PS==1的时候PDE直接指向物理页无PTE，低22位是业内偏移

线性地址只能拆成2段 10 ,22 ：大小为4mb 俗称大页

A位：是否被访问过，访问过置1

D位：是否被写过

3. 页目录表基址

(1)　拆分C0300000

1100 0000 00　　300 * 4 = C00

1100 0000 00　　300* 4 = C00

0000 0000 0000　0

经实验得知 C0300000 为CR3装载页目录表基址，系统通过这个地址找寻DirBase

(2)总结(系统通过0xC0300000这个线性地址找到DirBase)

①通过0xC0300000找到的物理页就是页目录表

②这个物理页即是页目录表本身也是页表

③页目录表是一张特殊的页表，每一项PTE不是普通的物理页，而是指向其他的页表

④如果我们要访问第N个PDE，那么有如下公式0xC0300000 + N*4

4. 页表基址

(1)　拆分C0000000

①　1100 0000 00　　300*4 = C00

0000 0000 00

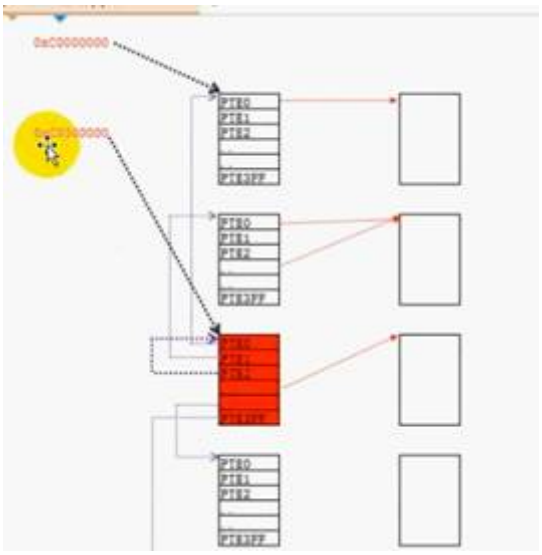
实验表明C0000000这个地址，找到的物理地址就是第一张PTT表所在地址

(2)拆分C0001000

①1100 0000 00　　300 * 4 = C00

00 0000 0001　　1 * 4 = 4

实验表明C0001000这个地址，找到的物理地址就是第二张PTT表所在地址



(3)总结：

①页表被映射到了从0xC0000000到0xC03FFFFF的4M地址空间

②在这1024个表中有一张特殊的表：页目录表

③页目录表被映射到了0xC0300000开始处的4K地址空间

注：页目录表和页表被映射到了C0000000到C03FFFFF的虚拟地址空间中，其中页目录表的地址为C0300000（它既是一个特殊的PTT,又是一个物理页），由这个地址的页目录表分别指向该空间中的其他PTT表

(4)应用

①什么是PDI与PTI　　10 10 12

②访问页目录表的公式　0xC0300000 + PDI*4

③访问页表的公式　　0xC0000000 + PDI*4096 + pti*4

5.29912分页

1.　为什么是2 -9 -9 -12

(1)　先确定了页的大小4kb，2^12 = 4096 = 4kb 所以12确定了

(2)　如果想增大物理内存的访问范围，就需要增大PTE,PTE越大能检索的范围也越大，再考虑对齐的因素，增加到8个字节

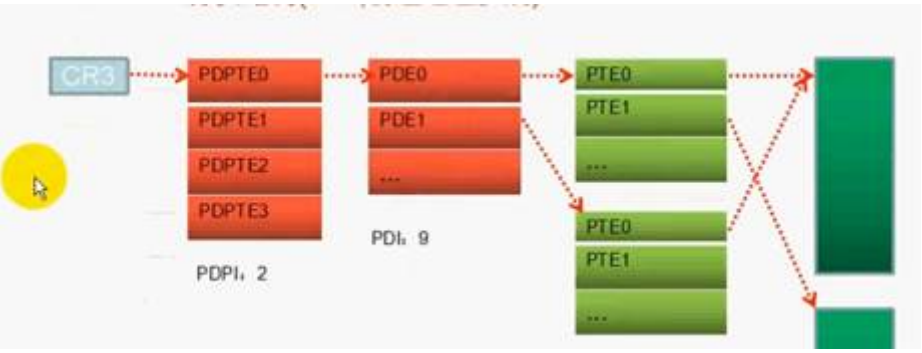
例32位 可检索 4G数据

33位 可检索 8G 数据

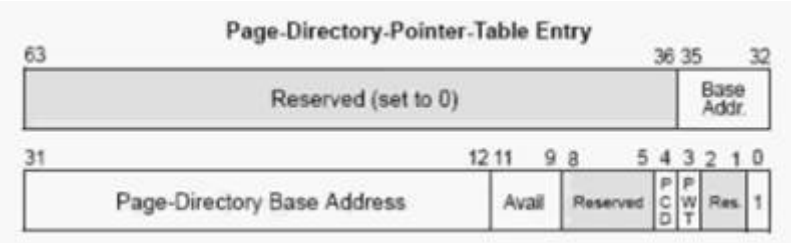
34位 可检索 16G数据

当PTE增加到8个字节，页大小4kb保持不变，则共计512种情况，2^9 = 512

PDE同理，也增加到8个字节

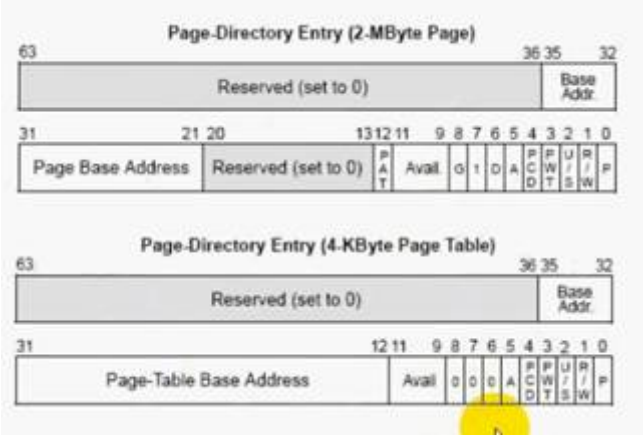


2.PDPTE



后12位补0，12-35位 PDT表地址

3.PDE结构



特别说明1.当PS = 1时是大页，35-21位是大页的物理地址，这样36位物理地址的低21位为0，这就意味着页的大小为2MB，且都是2MB对齐。

换句话说 32位寻址方式 被分成了 2 9 21 21位刚好是2MB

4.PTE结构

5.TLB

线性地址与物理地址的对应关系

系统读取4个字节的物理页，以2-9-9-12分页的模式，至少要读取24个字节，如果跨页，可能会读取更多。所以CPU内部做了一个表，来记录这些东西，这个表格是CPU内部的，和寄存器一样快，这个表格：TLB（Translation Lookaside Buffer）.

简单地说 TLB就是存取线性地址与物理地址对应关系的表格，由于每个进程都拥有自己的4gb空间，所以每个进程都有自己的TLB。

对于每个进程高2g内存，由于是共用系统内核的数据，所以在PDE与PTE中有一个G标志位，如果G位为1刷新TLB时将不会刷新PDE/PTE的G位为1的页，根据统计信息将不常用的地址废弃，最近最常用的保留

6. 中断与异常

(1)　什么是中断

①　中断通常是由CPU外部的输入输出设备（硬件）所触发的，供外部设备通知CPU，因此又叫中断请求

②　中断请求的目的是希望CPU暂时停止执行当前正在执行的程序，转去执行中断请求所对应的中断处理例程（中断处理程序在哪由IDT表决定）

③　可屏蔽请求与不可屏蔽请求

1)不可屏蔽中断

a.不受IF位影响，也就是说一旦发生，cpu必须处理。非可屏蔽中断位于IDT表中的2号位置

2)可屏蔽中断

a.在硬件级，可屏蔽中断是有一块专门的芯片来管理的，通常称为中断控制器。他负责分配终端资源和管理各个中断源发出的中断请求。为了便于标识各个中断请求，中断管理器通常用IRQ（Interrupt Request）后面加上数字来表示不同的中断

b.例如Windows系统中 时钟中断对应的中断请求 IRQ0

大多数操作系统时钟中断在10-100ms之间

c.可屏蔽中断如何处理？

每一个中断请求都对应着IDT表中中断号

例 0x30　　IRQ0　　时钟中断

(2)什么是异常

①异常通常是CPU在执行指令时检测到的某些错误，比如除0、访问无效页面等

②中断来自于外部设备，是中断源发起的，cpu是被动的

③异常来自于CPU本身，是CPU主动产生的

④INT N虽然被称为软件中断，但其本质是异常。EFLAG的IF位对INT N无效

(3)异常处理

①无论是由硬件设备触发的中断请求还是由cpu产生的异常，处理程序都在IDT表。

②例：缺页异常

比如 当物理页紧缺时，系统会将部分物理页暂时存储到硬盘中，从而这部分PTE的P位置0，当访问此类地址时，系统会产生缺页异常，进入异常处理函数，检查PTE，并将物理页置换回来

1.当PDE/PTE的P=0时

2.当PDE/PTE的属性为只读但程序试图写入时，一旦发生缺页异常，CPU会执行IDT表中0xE号中断处理程序，由操作系统来接管。

7. CPU缓存

1.CPU缓存是位于CPU与物理内存之间的临时存储器，它的容量比内存小的多，但是交换速度却比内存快很多

TLB:　　线性地址 <------> 物理地址

CPU缓存：物理地址 <-----> 内容