**电 子 科 技 大 学**

**实 验 报 告**

|  |  |
| --- | --- |
| **学生姓名：李逢君** | **学 号：2016060601010** |
| **一、实验室名称：主楼A2-413** | |
| **二、实验项目名称：虚拟内存综合实验** | |
| **三、实验原理：**  本实验通过用软件bochs配置Linux 0.11的虚拟运行环境，在Linux中运行一段无限循环的代码，通过bochs的调试工具修改代码中目标变量的内存中数据值使得程序退出无限循环，打印出结果。整个实验主要分两部分实施：逻辑地址转化为虚拟地址、虚拟地址转换为物理地址。 | |
| **四、实验目的：**  通过实验，掌握段页式内存管理机制，理解地址转换的过程 | |
| **五、实验内容：**  通过手工查看系统内存，并修改特定物理内存的值，实现控制程序运行的目的 | |
| **六、实验器材（环境配置）：**  Linux 内核（0.11）+ Bochs | |
| **七、实验步骤及操作：**  **1. 下载并安装bochs 2.5.1**  地址：bochs 2.5.1 <https://sourceforge.net/projects/bochs/files/bochs/2.5.1/>  **2. 准备**  2.1 运行本次实践提供的配置文件bochsrc-hd.bxrc以启动模拟一个linux(0.11)的环境    2.2. 命令行输入vi main.c 编辑如下文件    完成后退出  2.3. 命令行输入gcc -o main main.c 编译生成可执行文件main  2.4. ./main 执行，结果如下：    可以看到程序输出”the address of j is 0x3004”后进入死循环而无法退出程序运行，ctrl + c 退出死循环后关闭该窗口。  2.5. 右键点点击bochsrc-hd.bxrc，并选择debugger进入debug模式。  输入c    回车    在Display中运行./main    dubug窗口ctrl + c 可继续在断点处执行命令。  **3、逻辑地址转化为线性地址**   * **概念及数据结构** * **逻辑地址**   一个逻辑地址共48位，由两部分组成，段标识符(16)：段内偏移量(32)。段标识符是由一个16位长的字段组成，称为段选择符。其数据结构如下：     * **段描述符**   段选择符前13位是一个索引号，索引号指向段描述符，段描述符具体地址描述了一个段，多个段描述符发构成段描述符表，可以通过索引在段描述符表中找到一个具体的段描述符。而T1段代表段描述符表存放的位置，如果T1字段为 0，则表示该段描述符放在全局段描述符表(LDT)中，如果T1字段为1，则表示该段描述符放在局部段描述符表(GDT)中，最后2位涉及权限检查。其数据结构如下：     * **段描述符表的位置**   描述符表由系统地址寄存器(GDTR、IDTR、LDTR)指示其在物理存储器中的位置和大小，也就是说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IDTR寄存器中。   * **局部描述符表寄存器LDTR内容**   16位的LDTR并不直接定义LDT，它只是一个指向GDT中LDT描述符的选择符。   * **全局描述符表寄存器GDTR内容**   GDTR在物理存储器地址空间中定义全局描述符表GDT，其数据结构如下：    其中BASE指示GDT在物理存储器中开始的位置，LIMIT规定GDT的界限：   * **GDT/LDT/GDTR/LDTR关系图**      * **段选择符寄存器**   保护模式下CS、DS、SS、ES、FS、GS寄存器称为段选择符寄存器，其值不再是基址而是选择符，它从描述符表中选择一个定义存储器段大小和属性的描述符，其中DS为数据段寄存器，存放当前执行的程序所所用操作数的段地址。   * **具体步骤**  1. 通过LDTR寄存器找到LDT在GDT中的索引，即偏移。 2. 通过GDTR寄存器找到GDT的物理地址。 3. 通过GDT物理地址+LDT偏移可以计算出LDT记录在GDT的位置，里面有LDT的相关信息。 4. 通过信息的组合得到LDT的物理地址。 5. 通过DS数据段寄存器其在LDT的索引，即偏移。 6. 通过LDT的物理地址+DS偏移可以计算出DS记录在LDT的位置，里面有DS的相关信息。 7. 通过信息的组合得到DS的起始地址，其为LDT的段内偏移。 8. 将 LDT的物理地址（基址）+DS的起始地址（段内偏移）得到线性地址。  * **实现过程**   通过“sreg”命令可以查看局部描述符表寄存器LDTR的值     * **LDT索引值**   可以看到LDTR的值是ldtr:0x0068, dh=0x000082fa, dl=0x72d00068, valid=1，0x0068就是0000 0000 0110 1000，LDTR是一个16位的局部描述符寄存器，其指向GDT中LDT描述符的选择符。参照上述结构，选择符前13位是一个索引号，索引号指向段描述符。由于T1=0，所以高13位存放LDT在GDT中的索引值，即13，表示LDT放在GDT第14项。     * **GDT物理地址**   GDT的位置已经由GDTR给出gdtr:base=0x0000000000005cb8, limit=0x7ff，在物理地址0x0000000000005cb8。     * **LDT的段性质**   因此LDT的位置即我们要找的地址是0x00005cb8 + 13 \* 8，由于GDT表中的每一项占64位，即（8个字节），输入“xp /2w 0x00005cb8 + 13 \* 8”，该命令用来展示该位置的八个字节(w=word (4-bytes))。    得到0x0000000000005d20: 0x92d00068 0x000082fd，如果想确认是否准确，可以看 sreg输出中，ldtr所在行里，dl和dh的值，发现是一致的，故寻找正确。    由于为小端模式，所以0x000082fa 0x72d00068为LDT记录在GDT的描述信息，转化为二进制是  **0000 0000** 0000 0000 1000 0010 **1111 1101**  **1001 0010** **1101 0000** 0000 0000 0110 1000   * **LDT的物理地址**   上述描述信息对应下面的GDT性质    可以组合得到LDT的物理地址为0x00fd92d0。   * **DS的索引值**   前面已经提及DS存放当前执行的程序所所用操作数的段地址，而其在debug状态下其值不再是基址而是选择符，它从描述符表中选择一个定义存储器段大小和属性的描述符，故而需要通过选择符方式计算出其基地址。用”sreg”可以看到    ds:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=3，故ds=0000 0000 0001 0111  T1=1，表示查找LDT表，索引值为10（二进制）= 2（十进制），表示找LDT表中的第3个段描述符。   * **DS的起始地址**   因此从LDT的基地址出发查找，输入命令“xp /8w 0x00fd92d0”查看LDT表的前四项内容    得到  0x0000000000fd92d0: 0x00000000 0x00000000 0x00000002 0x10c0fb00  0x0000000000fd92e0: 0x00003fff 0x10c0f300 0x00000000 0x00fda000  第三项即：0x00003fff 0x10c0f300，可以用“sreg”输出中ds所在行的dl和dh值可以验证找到的描述符是否正确。    由于为小端模式，所以0x10c0f300 0x00003fff为LDT记录在GDT的描述信息，转化为二进制是  **0001 0000** 1100 0000 1111 0011 **0000 0000**  **0000 0000** **0000 0000** 0011 1111 1111 1111  合成的地址为0x10000000, 这就是DS在线性地址空间中的起始地址。   * **线性地址**   段基址+段内偏移，就是线性地址了。所以ds:0x3004的线性地址就是：0x10000000 + 0x3004 = **0x10003004**  用“calc ds:0x3004”命令可以验证这个结果。    **3、线性地址转化为物理地址**   * **概念及数据结构** * **线性地址**   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。  首先需要算出线性地址中的页目录号、页表号和页内偏移，它们分别对应了32位线性地址的10位+10位+12位，其数据结构如下：     * **控制寄存器**   存储器分页机制：CR0中的PG位设置为1，表示允许分页。CR3包含页目录基址，指向页目录的开头，如果发生缺页，则将发生缺页的地址保存在CR2中，控制寄存器的数据结构如下：  t1   * **具体步骤**  1. 通过线性地址得到页目录号、页表号和页内偏移。 2. 通过CR3找到页目录基址。 3. 通过页目录号计算页框的地址。 4. 通过页表号计算页表项的地址。 5. 通过页表项地址加上页内偏移得到变量的物理地址。 6. 修改该物理地址的值为全0，使死循环跳出，实验结束。        * **实现过程** * **页目录号、页表号和页内偏移**   根据线性地址的数据结构， 0x10003004页目录号是64，页号3，页内偏移是4。   * **页目录起始地址**   “creg”命令可以看到：    说明页目录表的基址为0。看看其内容，“xp /68w 0”：     * **物理页框号的地址**   页目录表和页表中的内容很简单，是1024个32位（正好是4K）数。这32位中前20位是物理页框号，后面是一些属性信息。由于线性地址0x10003004页目录号是64，所以第65个页目录项就是我们要找的内容，用“xp /w 0+64\*4”查看，得到：    故物理页框号为0x00fa6   * **物理页表项的地址**   页表所在物理页框号为0x00fa6，即页表在物理内存的0x00fa6000位置。由于线性地址0x10003004页号为3，所以从该位置开始查找3号页表项的地址，用“xp /w 0x00fa6000+3\*4”查看，得到     * **变量i的物理地址**   线性地址0x10003004对应的物理页框号为0x00fa5，和页内偏移0x004接到一起，得到0x00fa5004，这就是变量i的物理地址。  可以通过两种方法验证。  第一种方法是用命令“page 0x10003004”，可以得到信息：“linear page 0x10003000 maps to physical page 0x00fa5000”。    第二种方法是用命令“xp /w 0x00fa5004”，可以看到：    这个数值确实是main.c中i的初值   * **修改变量i的值**   通过直接修改内存来改变i的值为0，命令是：“setpmem 0x00fa5004 4 0”，表示从0x00fa5004地址开始的4个字节都设为0。然后再用“c”命令继续Bochs的运行，可以看到main退出了，说明i的修改成功了，实验完成。    C:\Users\PC\AppData\Roaming\Tencent\Users\543877815\QQ\WinTemp\RichOle\8$BY3~T59IHV[ZVD[}5XCWG.png | |
| **八、实验结论：**  通过手工查看系统内存，并修改特定物理内存的值，实现控制程序运行的目的 | |
| **九、总结及心得体会：**  通过实验，掌握段页式内存管理机制，理解地址转换的过程 | |
| **十、对本实验过程及方法、手段的改进建议：**  实验指导书和参考资料ppt的内容有点乱且有部分重复，如果能够整理后再给出就更好了。 | |
| **报告评分：**  **指导教师签字：** | |