**2.1 PPT 相关内容**

1. 实模式下的寻址方式以及实模式的缺陷

实模式下，段在内存中固定的位置

寻址方式：物理地址 = 段值\*16 +偏移

缺陷：通过改变段寄存器的值，我们可以随心所欲的访问内存任何一个单元，而丝毫不受到限制，不能对内存访问加以限制，也就谈不上对系统的保护

1. 保护模式下的寻址过程：

• 段寄存器中存储的是什么?

段选择子Selector（我们将要找的那个段描述符相对于数组首地址（也就是全局描述表的首地址）偏移位置）

GDT 是什么?

Global Descriptor Table,是一张存放Descriptor的表，可在全局内访问,所有进程想要访问全局可见的段时，从GDT查询，有且只有一个。进程从GDTR寄存器中获得GDT的位置，向它发起查询。

LDT 是什么?

LDT(Local)与GDT相同，但是不是全局的，对于某个进程，它只知道它自己的LDT。每个进程有自己的LDT，访问自己的段时从LDT查询。进程从LDTR寄存器中获得LDT的位置，向它发起查询。

如何区分LDT 和 GDT?

GDT包含系统使用的代码段、数据段、堆栈段和特殊数据段描述符，以及所有任务局部描述符表LDT的描述符。

LDT 和 GDT 的区别是什么?

在整个系统中，全局描述符表GDT只有一张(一个处理器对应一个GDT)，GDT可以被放在内存的任何位置，但CPU必须知道GDT的入口，也就是基地址放在哪里，Intel的设计者门提供了一个寄存器GDTR用来存放GDT的入口地址，程序员将GDT设定在内存中某个位置之后，可以通过LGDT指令将GDT的入口地址装入此寄存器。

LDT和GDT从本质上说是相同的，只是LDT嵌套在GDT之中。LDTR记录局部描述符表的起始位置，与GDTR不同LDTR的内容是一个段选择子。由于LDT本身同样是一段内存，也是一个段，所以它也有个描述符描述它，这个描述符就存储在GDT中，对应这个表述符也会有一个选择子，LDTR装载的就是这样一个选择子。LDTR可以在程序中随时改变，通过使用lldt指令。

如何定位到 Descriptor?

段选择子Selector+GDTR->描述符表->段描述符Descriptor

Descriptor 的内容有哪些?

包含段物理首地址、段界限、段属性，共64位

• GDTR 中的内容是什么?

GDTR用于表示GDT在内存中的段地址和段限（就是表的大小），因此GDTR是一个48位的寄存器，其中32位表示段地址，16位表示段限（最大 64K，每个描述符8字节，故最多有64K/8=8K个描述符）。

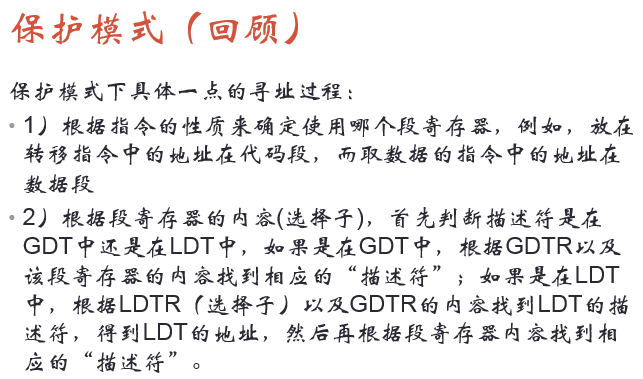
LDTR 中存储的是什么?

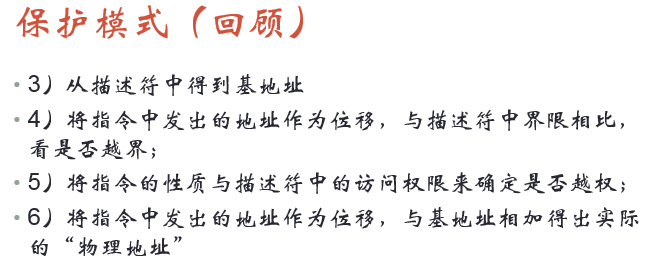
LDTR用于表示LDT在内存中的位置，但是因为LDT本身也是一种数据段，它必须有一 个描述符，且该描述符必须放在GDT中，因此LDTR使用了与DS、ES、CS等相同的机制，其中只存放一个“选择子”，通过查GDT表获得LDT的真正内存地址。

为什么 LDT 要放在 GDT 中?

GDT表只有一个，是固定的；而LDT表每个任务就可以有一个，因此有多个，并且由于任务的个数在不断变化其数量也在不断变化。

如果只有一个LDTR寄存器显然不能满足多个LDT的要求。因此intel的做法是把它放在放在GDT中。





3、选择子的作用：

• 选择子是什么？它的值存放在哪里？

• 选择子里面的内容有哪些？

• 为什么偏移地址大小是 13 位？

选择子是一个2字节的数，共16位，最低2位表示RPL（请求特权等级），第3位表示查表是利用GDT（全局描述符表）还是LDT（局部描述符表）进行，最高13位给出了所需的描述符在描述符表中的地址。（注：13位正好足够寻址8K项）

1. 描述符的作用：

保护模式下引入描述符来描述各种数据段

5、GDTR/LDTR 的作用：

• GDTR 的内容是什么？

• LDTR 的内容是什么？

GDTR用于表示GDT在内存中的段地址和段限（就是表的大小），因此GDTR是一个48位的寄存器，其中32位表示段地址，16位表示段限（最大 64K，每个描述符8字节，故最多有64K/8=8K个描述符）

LDTR用于表示LDT在内存中的位置，但是因为LDT本身也是一种数据段，它必须有一 个描述符，且该描述符必须放在GDT中，因此LDTR使用了与DS、ES、CS等相同的机制，其中只存放一个“选择子”，通过查GDT表获得LDT的真正内存地址。

1. 根目录区大小⼀定么? 扇区号是多少? 为什么？

不一定。扇区号=1+9\*2=19

根目录条目不一定。引导分区占一个扇区，每个fat占9个扇区

1. 数据区第⼀个簇号是多少? 为什么?

据区就起始于簇2。

在1.44M软盘上，FAT前三个字节的值必须是固定的，分别是0xF0、0xFF、0xFF，用于表示这是一个应用在1.44M软盘上的FAT12文件系统。本来序号为0和1的FAT表项应该对应于簇0和簇1，但是由于这两个表项被设置成了固定值，簇0和簇1就没有存在的意义了。

1. FAT 表的作用?

文件分配表被划分为紧密排列的若干个表项，每个表项都与数据区中的一个簇相对应，而且表项的序号也是与簇号一一对应的。

1. 解释静态链接的过程。

静态链接是由链接器在链接时将库的内容加入到可执行程序中的做法。

在编译main.c的时候，编译器还不知道printf函数的地址，所以在编译阶段只是将一个“临时地址”放到目标文件中，在链接阶段，这个“临时地址”将被修正为正确的地址，这个过程叫重定位。所以链接器还要知道该目标文件中哪些符号需要重定位，这些信息是放在了重定位表中。

在链接的时候，我们需要告诉链接器需要链接的目标文件和库文件（默认gcc会把标准库作为链接器输入的一部分）。链接器会根据输入的目标文件从库文件中提取需要目标文件。

知道了这些信息后，链接器就可以开始工作了，分为两个步骤：1）合并相似段，把所有需要链接的目标文件的相似段放在可执行文件的对应段中。2）重定位符号使得目标文件能正确调用到其他目标文件提供的函数。

1. 解释动态链接的过程。

所谓动态链接就是在运行的时候再去链接。

从动态库的角度来看，动态库像普通的可执行文件一样，有其代码段和数据段。为了使得动态库在内存中只有一份，需要做到不管动态库装载到什么位置，都不需要修改动态库中代码段的内容，从而实现动态库中代码段的共享。而数据段中的内容需要做到进程间的隔离，因此必须是私有的，也就是每个进程都有一份。因此，动态库的做法是把代码段中变化的部分放到数据段中去，这样代码段中剩下的就是不变的内容，就可以装载到虚拟内存的任何位置。那代码段中变化的内容是什么，主要包括了对外部函数和变量的引用。

动态库是把地址相关的内容放到了数据段中来实现地址无关的代码，从而使得动态库能被多个进程共享。

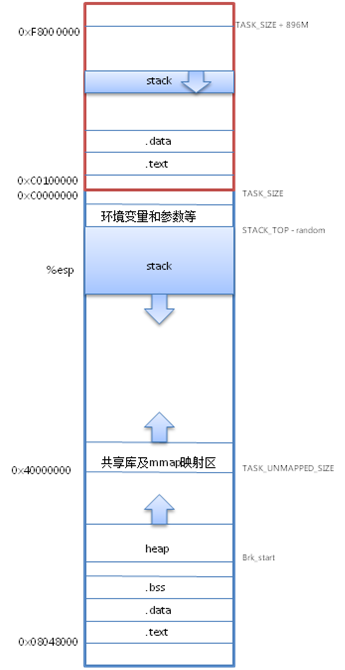
动态链接生成的可执行文件运行前，系统会首先将动态链接库加载到内存中。当所有的库都被加载进来以后，类似于静态链接，动态链接器从各个动态库中可以知道每个库都提供什么函数（符号表）和哪些函数引用需要重定位（重定位表），然后修正.got和.got.plt中的符号到正确的地址，完成之后就可以将控制权交给可执行文件的入口地址，从而开始执行我们编写的代码了。

1. 静态链接相关PPT 中为什么使用 ld 链接而不是 gcc。

用gcc的话有可能去调C库，使程序环境变得复杂，所以用ld

1. linux 下可执行文件的虚拟地址空间默认从哪里开始分配。

linux下，ELF可执行文件默认从地址0x08048000开始分配



**2.2 实验相关内容**

1. BPB 指定字段的含义

struct BPB {

u16 BPB\_BytsPerSec; //每扇区字节数

u8 BPB\_SecPerClus; //每簇扇区数

u16 BPB\_RsvdSecCnt; //Boot记录占用的扇区数

u8 BPB\_NumFATs; //FAT表个数

u16 BPB\_RootEntCnt; //根目录最大文件数

u16 BPB\_TotSec16; //扇区总数

u8 BPB\_Media; //介质描述符

u16 BPB\_FATSz16; //FAT扇区数

u16 BPB\_SecPerTrk; //每磁道扇区数

u16 BPB\_NumHeads; //磁头数

u32 BPB\_HiddSec; //隐藏扇区数

u32 BPB\_TotSec32; //如果BPB\_FATSz16为0，该值为FAT扇区数数

};

1. 如何进入子目录并输出 (说明方法调用)

根据子目录的开始簇号，计算出子目录的根目录开始的偏移量

int dataBase = BytsPerSec \* ( RsvdSecCnt + FATSz\*NumFATs + (RootEntCnt\*32 + BytsPerSec - 1)/BytsPerSec )

+(startClus-2)\*SecPerClus\*BytsPerSec;

然后递归调用打印文件的方法

1. 如何获得指定文件的内容, 即如何获得数据区的内容 (比如使用指针等)

数据区相对于文件的开端地址计算为：

int dataBase = BytsPerSec \* ( RsvdSecCnt + FATSz\*NumFATs + (RootEntCnt\*32 + BytsPerSec - 1)/BytsPerSec )

+(startClus-2)\*SecPerClus\*BytsPerSec;

即：每扇区字节数\*（Boot记录占用多少扇区+FAT扇区数\*FAT数目+（根目录最大文件数\*32+每扇区字节数-1）/每扇区字节数）+（当前文件簇号-2）\*每簇的扇区数\*每扇区字节数

然后调用fseek函数和fread函数，将文件内容读到content中

fseek(fat12,dataBase,SEEK\_SET);//得到文件指针

fread(content,1,512,fat12);

1. 如何进行 C 代码和汇编之间的参数传递和返回值传递

C代码和汇编之间通过栈来传递参数，返回值也是通过栈来传递

;第一个参数是输出类型，0——文件夹路径名 1——文件名

;第二个参数是输出内容

mov eax,[esp+4];栈中内容偏移四个开始存放参数

mov ecx,[esp+8]

1. 汇编代码中对 I/O 的处理方式, 说明指定寄存器所存值的含义

调用int 80h中断

mov eax, 4

mov ebx, 1

mov ecx,temp

mov edx, 1

int 80h

6. 可以要求解释某些看不懂的代码 (我看不懂的话, 你得讲给我听)