Lab2_questions

PPT内容

- 什么是实模式，什么是保护模式?

  > 实模式就是用基地址加偏移量就可以直接拿到物理地址的模式;
  >
  > 保护模式就是不能直接拿到物理地址的模式，需要进行地址转换

- 什么是选择子?

  > 选择子共 16 位，放在段选择寄存器里。段选择子是一个 16 位段描述符，指向了定义该段的段描述符表。
  >
  > 1. 第 0~1 位表示请求特权级，以什么样的权限去访问段
  > 2. 第 2 位 TL 表示选择 GDT 还是 LDT 方式，当 TL=0 时表示查询 GDT 表，TL=1 时则查找 LDT表
  > 3. 高 13 位表示在描述符表中的偏移（故描述符表的项数最多是 2 13）

- 什么是描述符?

  > 描述一个段是否在内存中；保护模式下引入描述符来描述各种数据段，起到地址转换的作用。
  >
  > - BASE：段基址，由上图中的两部分（BASE 31:24）和（BASE 23:16）组成
  > - G：LIMIT 的单位，该位为 0 表示单位是字节，为 1 表示单位是 4KB
  > - D/B：该位为 0 表示这是一个 16 位的段，为 1 表示这是一个 32 位段
  > - AVL：该位是用户位，可以被用户自由使用
  > - LIMIT：段的界限，单位由 G 位决定。数值上（经过单位换算后的值）等于段的长度（字节）–1
  > - P：段存在位，该位为 0 表示该段不存在，为 1 表示存在
  > - DPL：段权限
  > - S：该位为 1 表示这是一个数据段或者代码段，为 0 表示这是一个系统段（比如调用门，中断门等）
  > - TYPE：根据 S 位的结果，再次对段类型进行细分

- 什么是GDT，什么是LDT?

  > GDT：全局描述符表，是全局唯一的。存放一些公用的描述符，和包含各进程局部描述符表首地址的描述符。
  >
  > LDT：局部描述符表，每个进程都可以有一个。存放本进程内使用的描述符

- 请分别说明GDTR和LDTR的结构

  > GDTR：48 位寄存器，高 32 位放置 GDT 首地址，低 16 位放置 GDT 限长（限长决定了可寻址的大小）

> LDTR：16 位寄存器，放置一个特殊的选择子，用于查找当前进程的 LDT 首地址

- 请说明GDT直接查找物理地址的具体步骤。

> 1. 给出段选择子（放在段选择寄存器里）+ 偏移量
> 2. 若选择了 GDT 方式，则从 GDTR 获取 GDT 首地址，用段选择子中的 13 位做偏移，拿到 GDT中的描述符
> 3. 如果合法且有权限，用描述符中的段首地址加上 (1) 中的偏移量找到物理地址，寻址结束

- 请说明通过 LDT 查找物理地址的具体步骤。

> 1. 给出段选择子（放在段选择寄存器中）+ 偏移量
> 2. 若选择了 LDT 方式，则从 GDTR 获取 GDT 首地址，用 LDTR 中的偏移量做偏移，拿到 GDT中的描述符 1
> 3. 从描述符 1 中获取 LDT 首地址，用段选择子中的 13 位做偏移，拿到 LDT 中的描述符 2
> 4. 如果合法且有权限，用描述符 2 中的段首地址加上 (1) 中的偏移量找到物理地址，寻址结束

- 根目录区大小一定么？扇区号是多少？为什么？

> 不一定，需要根据目录项的最大个数计算
>
> 扇区号是 19 号，前面有 1 个引导扇区，2 个占 9 扇区的文件配置表（FAT1 和 FAT2）

- 数据区第一个簇号是多少？为什么？

> 每 12 位成为一个 FAT 项，代表一个簇。所以 2 个 FAT 项会占用 3 个字节。
>
> 在 1.44M 软盘上，FAT 前三个字节的值是固定的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，所以数据区就起始于簇 **2**

- FAT表的作用？

> 用来描述文件系统内存储单元的分配状态及文件内容的前后链接关系。
>
> 用来保存包含文件簇信息的表项，与数据区中的簇构成对应关系，实现文件的链式存储.

- 解释静态链接的过程

> 静态链接是在编译链接时直接将需要的执行代码拷贝到调用处。使用静态链接生成的可执行文件体积较大，包含相同的公共代码，造成浪费。
>
> • 空间与地址分配：扫描所有输入的目标文件，获取所有文件中节的信息，包括属性、长度和位置等，并且收集所有目标文件中的符号定义和符号引用信息。计算出输出文件中各个段合并后的长度与位置，并建立与目标平台程序地址空间的映射关系；
>
> • 符号解析：目标文件中定义和引用的符号，每个符号都对应于一个函数、全局变量或者静态变量。符号解析的目的就是将每个符号引用和一个符号定义关联起来；
>
> • 重定位：目标文件包含从地址 0 开始的代码和数据节。链接通过把每个符号定义与一个内存地址关联起来，从而重定位这些节，然后修改所有对这些符号的引用，使得它们指向这个内存位置。

- 解释动态链接的过程

> 使用动态链接的程序并不在一开始就完成动态链接，而是直到真正调用动态库代码（调用未被本文件实现的函数代码）时，载入程序才计算（被调用的那部分）动态代码的逻辑地址。这种方式使程序初始化时间较短，但运行期间的性能比不上静态链接的程序。
>
> • 动态链接器自举：动态链接器本身也是一个不依赖其他共享对象的共享对象，需要完成自举。
>
> • 装载共享对象：将可执行文件和链接器自身的符号合并成为全局符号表，开始寻找依赖对象。加载对象的过程可以看做图的遍历过程；新的共享对象加载进来后，其符号将合并入全局符号表；加载完毕后，全局符号表将包含进程动态链接所需全部符号。
>
> • 重定位和初始化：链接器遍历可执行文件和共享对象的重定位表，将它们 GOT/PLT 中每个需要重定位的位置进行修正。完成重定位后，链接器执行 .init 段的代码，进行共享对象特有的初始化过程（例如 C++ 里全局对象的构造函数）。
>
> • 转交控制权：完成所有工作，将控制权转交给程序的入口开始执行。

- 静态链接相关PPT中为什么使用ld链接而不是gcc?

> 为了避免 gcc 进行 glibc（即 C 运行库）的链接

- linux下可执行文件的虚拟地址空间默认从哪里开始分配。

> 0x08048000

实验内容

- BPB指定字段的含义

名称	开始字节	长度	内容	参考值
BS_jmpBOOT	0	3	一个短跳转指令	jmp LABEL_START nop
BS_OEMName	3	8	厂商名	'ForrestY'
BPB_BytesPerSec	11	2	每扇区字节数	0x200
BPB_SecPerClus	13	1	每簇扇区数	0x1
BPB_ResvdSecCnt	14	2	Boot 记录占用多少扇区	0x1
BPB_NumFATs	16	1	共有多少 FAT 表	0x2
BPB_RootEntCnt	17	2	根目录区文件最大数	0xE0
BPB_TotSec16	19	2	扇区总数	0xB40
BPB_Media	21	1	介质描述符	0xF0
BPB_FATSz16	22	2	每个 FAT 表所占扇区数	0x9
BPB_SecPerTrk	24	2	每磁道扇区数	0x12
BPB_NumHeads	26	2	磁头数（面数）	0x2
BPB_HiddSec	28	4	隐藏扇区数	0
BPB_TotSec32	32	4	如果BPB_TotSec16是 0，则由这个值记录扇区数	0
BS_DrvNum	36	1	中断 13 的驱动器号	0
BS_Reserved1	37	1	保留，未使用	0
BS_BootSig	38	1	扩展引导标记（29h）	0x29
BS_VolID	39	4	卷序列号	0
BS_VolLab	43	11	卷标	OrangeS0.02
BS_FileSysType	54	8	文件系统类型	'FAT12'
引导代码及其他内容	62	448	引导代码及其他数据	引导代码（剩余空间用0填充）
结束标志	510	2	0xAA55	0xAA55

- 如何进入子目录并输出（说明方法调用）

  > 采用递归，先读取当前目录的子目录，再递归进入各子目录递归读取。

- 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）

  > 首先计算出数据区的起始地址，再根据文件目录项的 DIR_FstClus（开始簇号）字段，计算出偏移量，去数据区中读取相应的内容。对于大于 512 字节的文件，需要在 FAT 表中查找下一簇的簇号重复计算数据区地址和读取的过程，直至遇到坏簇或读取结束。

- 如何进行C代码和汇编之间的参数传递和返回值传递

  > 参数被依次压入栈中，在汇编中根据 esp 计算参数在栈中的位置；返回值存在寄存器 eax 中

- 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

  > 通过 esp 计算传入的参数位置，eax 存系统调用号，ebx 存文件描述符，ecx 存起始地址，edx 存长度