|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 武汉大学国家网络安全学院教学实验报告 | | | | | |
| 课程名称 | 操作系统设计与实践 | | 实验日期 | | 2024.10.14 |
| 实验名称 | 由盘上结构实现程序加载 | | 实验周次 | | 第五周 |
| 姓名 | 学号 | | 专业 | | 班级 |
| 王亚鹏 | 2022302181161 | | 信息安全 | | 5班 |
| 杨依磊 | 2022302181159 | | 信息安全 | | 5班 |
| 杜泓波 | 2022302181162 | | 信息安全 | | 5班 |
| 侯名扬 | 2022302181165 | | 信息安全 | | 5班 |
| 1. 实验目的及实验内容   （要求掌握的知识；实验内容；原理分析） | | | | | |
| 总目标：  如何从软盘读取并加载一个Loader程序到操作系统，然后转交系统控制权  实验内容：  1.向软盘镜像文件写入一个你自己任意创建的文件，手工方式在软盘中找到指定的文件，读取其扇区信息，记录你的步骤。  2.将指定的可执行文件装入指定内存区，并执行，记录原理与步骤  3.学会使用xxd读取二进制信息，通过1、2来验证。  原理分析：  1. 向软盘镜像文件写入文件并手工查找文件扇区  **FAT12文件系统：**  FAT12是一种文件系统，常用于软盘中。它的结构包括引导扇区、文件分配表（FAT表）、根目录区域和文件数据区域。  引导扇区：位于磁盘的第一个扇区，包含文件系统的元数据和引导代码。  FAT表：记录每个文件的簇链（文件数据存放的簇集合）。  根目录区域：包含文件的元数据（文件名、大小、起始簇号等）。  数据区域：存储实际的文件内容，按簇存放。    **文件的写入：**  当文件被写入软盘时，操作系统会根据FAT12文件系统的规则，将文件的内容分配到多个簇（如果文件较大），并在FAT表中记录这些簇的链表。  **手工查找文件扇区：**  使用十六进制编辑器（如 xxd）查看软盘镜像时，可以解析FAT表和根目录表，找到文件对应的起始簇号，然后根据起始簇号找到文件的具体数据扇区。  根目录中每个文件都有一个条目，条目中包含文件的起始簇号。根据该簇号，可以从FAT表中找到该文件的完整簇链，最终找到所有存储文件的扇区。    2. 将可执行文件装入指定内存区并执行  **引导扇区加载：**  在软盘中，系统启动时BIOS会读取软盘的第一个扇区（即引导扇区，512字节），并将其加载到内存地址 0x7C00。如果该扇区最后两个字节是 0xAA55（引导签名），BIOS会将控制权交给该扇区中的程序，即Loader。  **Loader程序：**  Loader通常是一个汇编语言写的小程序，负责进一步加载操作系统或其他可执行程序。它需要遵循BIOS的规范，在0x7C00地址开始执行。编写简单的Loader程序，通常包括读取其他扇区的代码并加载到内存的指定位置。编译该Loader程序为二进制格式，并将其写入软盘镜像的引导扇区。BIOS将会在系统启动时，将软盘引导扇区的Loader代码加载到内存并执行。  关键点：  BIOS将控制权交给Loader程序的过程实际上是硬件对软盘内容的自动读取和跳转。这个Loader可以通过调用BIOS中断（如 INT 13h）读取其他软盘扇区内容，并加载更多数据或启动操作系统。  总结：  Loader程序的设计需要遵循x86架构和BIOS启动的规范。通过在软盘的引导扇区写入Loader程序，控制权可以在启动时交给Loader，由它加载并执行更多操作。  3. 使用 xxd 读取二进制信息验证  **xxd工具：**  xxd 是一种二进制文件查看工具，可以将二进制文件转换为十六进制显示，便于分析和验证数据。  将Loader程序写入软盘镜像后，可以使用 xxd 检查镜像文件的前512字节（引导扇区）是否包含了正确的Loader代码。  通过检查0x7C00处的内容，确保Loader程序被正确加载并包含期望的代码。通过 xxd 查看软盘镜像，可以验证FAT表、根目录表是否正确生成，文件是否被正确写入。还可以手动解析每个文件的起始簇号，确认其在磁盘中的存储位置。 | | | | | |
| 1. 实验环境及实验步骤   （本次实验所使用的器件、仪器设备等的情况；具体实验步骤） | | | | | |
| 实验环境：  **X86 32位Linux环境**  实验步骤：  **1. 向软盘镜像文件写入一个你自己任意创建的文件，手工方式在软盘中找到指定的文件，读取其扇区信息，记录你的步骤。**    ① 创建虚拟软盘data.img   ② 在FreeDos中进行格式化（FAT12）   ③ 将data.img挂载到Linux中，并写入文件    **修改bochsrc 配置文件，freedos.img 作为启动软盘（A盘），将软盘data.img作为B盘插入**          **查阅资料得，edit操作被封装在c.img虚拟硬盘中。而a.img也就是freedos.img并没有edit操作。修改borcsrc，使c.img作为启动盘**            **使用xxd二进制查看器查看data.img的偏移0x2600处**    注意偏移为0x1A的内容，为0002，代表此条目对应的开始簇号。  又在本FAT12中，因为BPB\_RootEntCnt=0xE0=14\*16+0=244，即条目最多为244个，又因为每个条目占用32个字节，故244\*32/512=14，即该根目录区占14个扇区。因此数据区开始扇区号= 根目录区开始扇区号+ 14 = 33，可得偏移为512\*33=0x4200，用xxd查看此处。    **2. 将指定的可执行文件装入指定内存区，并执行，记录原理与步骤**  为了变成能够被DOS以及 Linux识别的软盘，必须给引导扇区增加BPB块    使用bximage.exe命令生成a.img, 刚生成的a.img文件内容都是0x00,    nasm boot.asm -o boot.bin  dd if=boot.bin of=a.img bs=512 count=1 conv=notrunc  把生成的boot.bin写入磁盘引导扇区，使用[虚拟机](https://so.csdn.net/so/search?q=%E8%99%9A%E6%8B%9F%E6%9C%BA&spm=1001.2101.3001.7020" \t "_blank)执行仍然可以成功，并且现在的软盘已经能够被DOS以及Linux识别了，我们已经可以方便地往上添加或删除文件了。    <bochs:1> **b 0x7c00** ←*在开始处设置断点* <bochs:2> **c** ←*执行到断点* | | | | | |
| 断点应该设在0x7cb4处即第114行程序执行到那里时es:di为0x9000:0x12b即内存0x9012b处容易发现这里正是Loader.bin  首先我们从根目录区中的Loader.bin的条目，获取文件的起始扇区号，然后加上BPB\_RsrvSecCnt+BPB\_FATSz16\*2-2+RootDirSectors=1+（9\*2）+14-2=31,，其中DeltaSectorNo=BPB\_RsrvSecCnt+BPB\_FATSz16\*2-2=17。得到的结果才是文件的实际的起始扇区。获得起始扇区后，我们就可以调用ReadSector来读取扇区了。然后从FAT1表中获取FATEntry的值，判断是否为0FFFh，如果是，结束加载；如果不为0FFFh，意味着该文件没有读取完成，需要读取下一个扇区，此时的FATEntry的值，就是下一个扇区号，再将其转换为实际扇区号，再进行读取。 | | | | | |
| 1. 实验过程分析   （详细记录实验过程中发生的故障和问题，进行故障分析，说明故障排除的过程及方法。根据具体实验，记录、整理相应的数据表格等） | | | | | |
| **故障及其排除方法**：无 | | | | | |
| 1. 实验结果总结   （对实验结果进行分析，完成思考题目，并提出实验的改进意见） | | | | | |
| **思考题目：**  **1. FAT12格式是怎样的？**  FAT12（File Allocation Table 12）是最早期的文件系统之一，常用于软盘和小容量存储设备。它的结构主要包括以下几个部分：   * 引导扇区（Boot Sector）：包含引导程序和文件系统信息，如每扇区的字节数、保留扇区数、FAT表数、根目录最大文件数等。 * FAT表（File Allocation Table）：用于跟踪文件在磁盘上的存储位置。FAT12使用12位地址，最多可以支持 $2^{12}$（即4096）个簇。 * 根目录区（Root Directory Area）：存放根目录下的文件信息，包含文件名、扩展名、起始簇号、文件大小等信息。 * 数据区（Data Area）：存放实际文件数据。   FAT12的扇区和簇的大小通常是512字节，且根目录的大小是固定的。由于其结构简单，FAT12适合小容量的存储设备。  **2. 如何读取一张软盘的信息？**  要读取一张软盘的信息，可以按照以下步骤进行：  使用适当的工具：  在Linux上，可以使用命令如 dd 来读取软盘镜像。例如，dd if=/dev/fd0 of=floppy.img 将软盘内容写入 floppy.img 文件。  分析引导扇区：  读取软盘的第一个扇区（引导扇区），其中包含文件系统的基本信息。  使用工具（如 xxd 或 hexdump）查看引导扇区的内容，分析其中的参数。  读取FAT表：  根据引导扇区提供的信息，确定FAT表的位置和大小，然后读取FAT表。  读取根目录和数据区：  根据引导扇区的定义，读取根目录区域和数据区中的文件。  **3. 如何在软盘中找到指定的文件？**  要在软盘中找到指定的文件，可以按照以下步骤：  解析根目录：  从根目录读取文件项，检查每个文件项的文件名和扩展名，找到目标文件。  使用FAT表：  如果目标文件是子目录，查找其起始簇号，并根据FAT表的链表结构跟踪文件的存储位置。  读取文件内容：  找到目标文件的起始簇后，读取其对应的簇数据，直到文件末尾。  **4. 如何在系统引导过程中，从读取并加载一个可执行文件到内存，并转交控制权？**  在系统引导过程中，通常会执行以下步骤：  引导程序加载：  当计算机启动时，BIOS从软盘的引导扇区读取引导程序并加载到内存。  读取文件：  引导程序解析FAT表，找到目标可执行文件的位置，读取该文件到指定的内存地址。  设置堆栈：  在加载可执行文件之前，设置程序的堆栈，确保正常执行。  转交控制权：  在加载程序执行之前，将CPU控制权转交给可执行文件的入口点（通常是在文件的头部指定）。  **5. 为什么需要这个Loader程序不包含DOS系统调用？**  Loader程序不包含DOS系统调用的原因主要有：  环境独立性：Loader程序是在操作系统加载之前执行的，因此不依赖于任何操作系统的功能。  简化性：通过避免使用系统调用，Loader程序可以更加简化，从而减少潜在的错误和不必要的复杂性。  底层访问：Loader程序需要直接访问磁盘和内存，使用系统调用会增加不必要的抽象层。  **6. 为什么在前面几个章节中a.img，不能直接mount，在本章代码里面却可以？**  在前面几个章节中，a.img未按照标准文件系统格式（如FAT12、FAT32等）创建的，因此系统无法直接识别和挂载它。而在本实验中，代码实现了对特定文件系统格式的解析和处理，能直接读取和加载 a.img 中的数据，使得能够“挂载”这个镜像。  **7. 扩展提高：调研在硬盘上，文件系统格式为FAT32或者NTFS，应该怎么来实现类似功能呢？**  对于FAT32和NTFS，可以采取以下步骤来实现类似的功能：  FAT32：  FAT32与FAT12类似，但支持更大的存储设备和更大的文件。首先，读取引导扇区，解析FAT32的结构。  FAT32使用32位地址，因此需要更新解析方法以适应新的簇编号。  通过FAT表找到目标文件的簇，并读取数据区内容。  NTFS：  NTFS比FAT复杂，采用更高级的结构，包括MFT（主文件表）来存储文件的元数据。  读取引导扇区以获取NTFS引导信息。  解析MFT以找到目标文件的信息，然后根据指向的簇读取实际文件数据。NTFS支持更丰富的功能，如访问控制、压缩等，因此实现时需考虑这些特性。  **改进意见：**   **软盘镜像文件的操作自动化** 实验的第一步手动在软盘镜像中查找文件、读取扇区信息较为繁琐，可以考虑使用更自动化的工具来提升效率。例如，使用Python脚本或其他文件系统分析工具（如mtools或floppyimage）来自动读取镜像文件并获取扇区信息。通过这些工具，能够加快文件操作的速度，并减少手动出错的可能性。   **可执行文件的加载与内存映射优化** 在实验的第二步中，手动将指定的可执行文件装入内存并执行，虽然能很好理解操作系统加载机制，但可以通过增加调试工具（如gdb）进行更细致的调试与观察，深入了解内存中的每一个步骤。此外，使用更复杂的可执行文件或自定义链接器脚本来控制程序在内存中的布局，能够更好理解内存分配和段结构。   **更深入学习二进制文件格式和结构** 在使用xxd读取二进制文件信息时，可以结合分析特定的文件格式（如FAT12文件系统或ELF/PE格式），不仅查看数据，还要理解其结构。增加对文件格式的深入学习，可以帮助在实验过程中更准确地定位数据、扇区和内存块。例如，在实验中，读取FAT12文件分配表或分析ELF可执行文件的头部信息，可以提升对文件系统和二进制文件的理解。   **实验反馈循环与结果验证** 实验的1和2步骤可以通过自动化工具（如脚本或虚拟机环境）重复验证。在每次对软盘镜像进行操作后，能够立刻通过脚本自动化验证结果，确保每一个操作都能正确读取或写入。在实验过程中保持一个反馈循环，能更快发现错误并作出调整。结合调试器，可以在执行可执行文件的每一个步骤中通过断点检查寄存器、内存、标志位等信息，以确保加载执行过程的正确性。   **错误处理与异常情况分析** 增加对软盘文件写入与加载过程中可能出现的异常情况的处理。例如，在软盘镜像中写入文件时，可能会遇到文件损坏、磁盘空间不足等问题。可以在实验中增加对这些错误的分析和处理机制，以应对实验中可能的异常情况，提高系统的健壮性。   **扩展到现代系统环境下的仿真与实践** 现代操作系统已经不再使用软盘作为主要的引导设备，但引导加载器的原理仍然十分重要。可以将实验中的软盘引导程序扩展到现代设备上，例如USB引导或硬盘引导。通过使用如QEMU等虚拟机模拟硬件环境，能够在不依赖于老旧软盘驱动的情况下进行加载器相关实验，从而获得更广泛的实用性。 | | | | | |
|  | | | | | |
| 1. 各人实验贡献与体会（每人各自撰写） | | | | | |
| 实际上每个人都各自独立地完成了实验  1.王亚鹏：全程参与实验，完成了汇编代码调试（尤其是遍历验证loader程序的位置）以及思考题的回答  **1. 对文件系统的深入理解**  通过创建和操作软盘镜像，学会了如何从镜像文件中手动查找文件的扇区信息，理解了FAT12文件系统中簇、扇区的关系以及文件分配表（FAT）的运作方式。在使用 xxd 等工具查看二进制信息时，能够清楚地看到文件在存储中的表现形式，这是一个查看内存内容非常有用的命令  **2. 学习Loader的工作原理**  Loader是引导操作系统启动的核心组件。将Loader程序加载到内存中，并通过手动操作让系统转交控制权给Loader。通过分析系统启动时从软盘读取数据的过程，了解到Loader程序不依赖DOS系统调用是因为它需要直接控制硬件，而不是通过操作系统的中间层  **3. 存在的一些挑战**  最大困难是在FAT12的扇区和簇计算上，特别是如何将软盘中的二进制数据与文件系统结构对应起来。反复使用 xxd 工具分析二进制数据，逐渐掌握了如何根据目录项来推断文件的物理位置。此外，下断点的实际位置也需要去手动计算，十分的不方便  2.杨依磊：全程参与实验  本次实验围绕操作系统中从软盘读取文件并加载程序的过程展开，涉及FAT12文件系统的基本原理、引导扇区、加载器的设计与实现等。 1. FAT12文件系统的学习在本次实验中，我首先学习了FAT12文件系统的结构。FAT12是一种用于软盘的文件系统，它通过扇区、簇、分区等概念来管理文件的存储和读取。通过实验，我了解了根目录区的条目格式、FAT项的格式以及如何通过这些信息读取软盘中的文件。2. 手动读取软盘文件的过程实验要求我们手动写入一个文件到软盘镜像，并手动查找该文件的扇区信息。 这一过程需要理解软盘镜像的结构，并通过工具如 xxd 来查看二进制数据，找到文件的具体存储位置。通过这一过程，我了解了操作系统在底层如何通过精确的地址定位来加载文件，进而执行程序。 3. 加载器的实现与系统控制权的转移实验的核心是实现一个加载器 ，将可执行文件装入内存并转交系统控制权。通过这部分实验，我理解了加载器的作用以及在引导过程中如何读取一个可执行文件，并将其加载到指定的内存区域。值得注意的是，这个加载器并不依赖于DOS系统调用，而是通过操作系统本身的引导逻辑直接运行。这一过程让我认识到操作系统引导时的自包含性  3.杜泓波：全程参与实验  本次实验的总体目标是通过软盘读取并加载一个Loader程序到操作系统，并转交系统控制权。在实验过程中，学习了如何手动操作软盘镜像文件、加载可执行文件到指定内存区域，并利用xxd等工具分析和验证二进制数据。整个过程让我对系统底层的加载与执行机制有了更深入的理解，也加深了我对软盘文件系统和引导加载器的认识。  **1. 向软盘镜像文件写入并读取文件**  实验的第一步是向软盘镜像写入一个文件，并手动找到这个文件的扇区信息。在实际操作中，我使用了工具（如dd）来向软盘镜像写入文件，并通过分析FAT12文件系统结构手动查找文件所在的扇区。通过这一步，我理解了软盘文件系统的组织方式，包括FAT表的工作原理以及数据区的分配方式。  在这一过程中，我意识到文件的扇区查找并不只是通过简单的目录结构，还需要解析FAT表中的簇链，以了解文件的分布情况。这让我更深入理解了文件系统的基础概念，尤其是对于FAT12这种较为古老的文件系统，它的数据管理和现代系统有明显的区别。  **2. 将可执行文件装入指定内存并执行**  第二步是将一个可执行文件加载到指定的内存区域，并进行执行。这个过程模拟了操作系统引导加载器的功能，即从磁盘读取可执行文件并将其放入内存，然后转交控制权。在实验中，我使用了BIOS的中断功能（如int 13h）来从软盘读取文件，并通过内存复制将文件加载到目标位置。  通过调试工具，我能够观察到程序的加载和执行过程，并监控寄存器和内存的变化。通过这一步骤，我对操作系统的启动过程有了更加直观的认识，理解了BIOS、引导加载器和操作系统之间的工作衔接。  **3. 使用xxd工具验证二进制信息**  在第三步中，我使用xxd工具来读取二进制文件信息，验证软盘中的文件数据是否正确写入，以及可执行文件在内存中的布局是否符合预期。通过对比加载前后的二进制数据，我能够确认文件读取、写入及执行的准确性。  xxd不仅仅是一个查看二进制文件的工具，还可以通过数据分析帮助理解文件格式和内存结构。在实验中，通过对比软盘镜像中的文件数据和加载到内存中的可执行文件，我验证了每一个步骤的正确性。这种验证方式不仅直观，还增强了我对底层数据操作的信心。  **实验的收获与理解**  本次实验让我对计算机系统启动时的底层操作有了全面的理解，从软盘镜像文件的操作到可执行文件的加载和执行，整个过程模拟了操作系统的引导流程。通过实验，我学会了如何在软盘中定位文件、如何手动加载文件到内存，并通过二进制工具验证实验的结果。  实验还让我认识到现代系统与旧式引导方式之间的区别，尤其是在引导加载器和文件系统方面的不同。虽然软盘和FAT12已不再常用，但这些原理仍然是理解现代操作系统引导过程的基础。  4.侯名扬：全程参与实验  在此次实验中，我们的主要任务是通过软盘读取并实现一个Loader程序的加载，进而将系统的控制权交接给该程序。  1. 在软盘镜像中写入并读取文件 实验的第一阶段，我尝试向软盘镜像中写入文件，并手动搜索该文件所在的扇区。我运用了如dd等工具完成了文件的写入，并通过深入分析FAT12文件系统结构，手动定位到文件所在的扇区。这个过程使我明白了软盘文件系统的构造原理，包括FAT表的运行机制和数据区的分配方法。 2. 将可执行文件加载至指定内存并执行 第二阶段，我尝试将一个可执行文件导入指定的内存区域并执行。这个过程模拟了操作系统引导加载器的工作，即从磁盘读取可执行文件并将其导入内存，最后交出控制权。在实验中，我运用BIOS中断（如int 13h）从软盘读取文件，并通过内存复制将文件导入目标位置。 3. 利用xxd工具验证二进制信息 在第三阶段，我使用xxd工具读取二进制文件信息，以验证软盘中的文件数据是否正确写入，以及可执行文件在内存中的布局是否满足预期。通过对比加载前后的二进制数据，我确认了文件读取、写入和执行的准确性。 此次实验使我学会向软盘镜像文件写入一个任意创建的文件，手工方式在软盘中找到指定的文件，读取其扇区信息，记录你的步骤，将指定的可执行文件装入指定内存区，并执行，记录原理与步骤 ，最后我还学会使用xxd读取二进制信息 | | | | | |
| 1. 教师评语   （实验报告的考评：依据实验内容完整度、实验步骤清晰度、实验结果与分析正确性、实验心得与思考的全面性、实验报告文档的规范性等五个维度综合考评）   |  |  | | --- | --- | | 85-100 | * 实验内容完整或者有超出课程实验大纲的内容； * 实验步骤详尽，能够体现完整的实验过程； * 实验结果正确且实验数据分析得当； * 实验心得与思考全面并且有自己的独立思考； * 实验报告文档规范、排版整齐。 | | 75-84 | * 实验内容较为完整； * 实验步骤较为详尽，能够体现实验过程； * 实验结果正确且实验数据分析较为得当； * 实验心得与思考全面； * 实验报告文档规范、排版较为整齐。 | | 60-74 | * 实验内容有缺失； * 实验步骤不够详尽，不能够体现完整的实验过程； * 实验结果部分正确； * 实验心得与思考无或者不够深入； * 实验报告文档规范性有待增强。 | | 60以下 | * 实验内容严重缺失、实验态度不够端正 * 实验步骤不够详尽，不能够体现完整的实验过程； * 实验结果部分正确； * 实验心得与思考无或者不够深入； * 实验报告文档规范性有待增强。 | | | | | | |
|  | | | | | |
| **教师评分（请填写好姓名、学号）** | | | | | |
| 姓名 | | 学号 | | 分数 | |
| 王亚鹏 | | 2022302181161 | |  | |
| 杨依磊 | | 2022302181159 | |  | |
| 杜泓波 | | 2022302181162 | |  | |
| 侯名扬 | | 2022302181165 | |  | |
| 教师签名：  年 月 日 | | | | | |