EOS 操作系统实验教师参考

英真时代 编著

Engintime 北京英真时代科技有限公司

EOS 操作系统实验教师参考

英真时代 编著

版本 2.1

北京英真时代科技有限公司

内容简介

本书是操作系统集成实验环境 OS Lab 和《EOS 操作系统实验教程》的教师参考书。本书主要分为两部分：第一部分可以帮助教师将 OS Lab 和 EOS 操作系统应用于高校操作系统原理课程，以及相应的实验教学；第二部分提供了所有配套实验的答案和注意事项等内容，供教师参考。

北京英真时代科技有限公司联系方式

地址：北京市房山区拱辰大街 98 号 7 层 0825

邮编：102488

电话：010-60357081

QQ：964515564

邮箱：support@tevation.com

网址：http://www.engintime.com

目 录

第 1 章 概述

本章内容可以帮助教师将操作系统集成实验环境 OS Lab 和 EOS 操作系统应用于高校操作系统原理课程，以及相应的实验教学。有关 OS Lab 在实验室部署的方法和基本使用方法请参见《Engintime OS Lab 安装与使用指南》。有关 OS Lab 和 EOS 操作系统的详细介绍请参见配套教材《EOS 操作系统实验教程》。

1.1 配套资料

在操作系统实验教学的各个环节，OS Lab 都提供了完善的配套资料。这些资料包括文档、源代码、幻灯片、培训录像等。在这些配套资料的帮助下，教师可以方便、灵活的组织学生开展操作系统实验。

所有配套资料都可以从 OS Lab 的产品光盘中获得。详细信息可以参见表 1-1。

序号	资料名称	资料说明	使用者
1	**《EOS 操作系统实验教程》**	该文档是使用 OS Lab 进行操作系统实验的核心文档。该文档结合操作系统原理详细分析了 EOS 操作系统的架构和源代码，并包含有若干个实验题目供教师选用。	教师 学生
2	**《EOS 操作系统实验教师参考》**	该文档可以帮助教师将EOS操作系统和OS Lab应用于高校操作系统原理课程，以及相应的实验教学。	教师
3	**EOS 操作系统实验学生包**	包含有若干子文件夹，每个子文件夹对应一个实验题目。在子文件夹中包含了完成实验题目所需的源代码文件等。	教师 学生
4	**EOS 操作系统实验教师包**	包含有若干子文件夹，每个子文件夹对应一个实验题目。在子文件夹中除了包含有学生包中的内容外，还包含有供教师使用的源代码参考答案，幻灯片等。教师可以在向学生布置实验，或者检查实验结果时使用。	教师
5	**《Engintime OS Lab 安装与使用指南》**	该文档用于帮助教师或者实验室管理员在各种实验室环境下顺利的安装 OS Lab，并开始使用 OS Lab。	教师 实验室管理员
6	**Engintime OS Lab 帮助**	该文档随 OS Lab 一起被安装到计算机中。在启动 OS Lab 后，可以随时按 F1 键打开该文档获得帮助。	教师 学生
7	**其它**	其它文档会随 OS Lab 一起被安装到计算机中。包括辅助工具的帮助文档，以及技术参考文档等。	教师 学生

表 1-1：配套资料清单

1.2 向学生分发配套资料

教师可以从 OS Lab 的产品光盘中获得所有配套资料，如果要将 OS Lab 应用于实际的教学中，还应该将必要的资料分发给学生。在将配套资料分发给学生时，教师可以根据不同的情况采用下面的两种方式之一：

1. 最简单的方式是将《EOS 操作系统实验教程》电子版文件分发给学生。在该教程的前言部分会引导学生从互联网下载所有其它必要的配套资料。当然，如果教师已经采用了该教程的纸制教材，就更方便了。
2. 如果师生连接互联网不是很方便，可以采用另外一种方式。教师可以将：
   - 《EOS 操作系统实验教程》电子版
   - EOS 操作系统实验学生包
   - OS Lab 演示版安装包（同样包含在 OS Lab 产品光盘中）

   一起分发给学生。学生可以使用 OS Lab 演示版安装包在自己的计算机上免费安装和使用 OS Lab 演示版（与正式版相比，在功能上会有一些限制），学生可以使用演示版阅读 EOS 源代码，并对 EOS 源代码进行简单的调试，从而完成对实验的预习和复习。

在分发配套资料时请教师注意：

- 请不要将《EOS 操作系统实验教师参考》和“EOS 操作系统实验教师包”分发给学生，以免影响实验效果。
- 由于所有配套资料都有显式的版权和授权声明，所以请不要在公共域（例如互联网）中分发任何配套资料。

1.3 配套实验清单

提供了 12 个配套实验供教师选用，请参见表 1-2。

序号	实验名称	实验目的	实验性质	建议学时
1	**实验环境的使用**	● 熟悉操作系统集成实验环境 OS Lab 的基本使用方法。 ● 练习编译、调试 EOS 操作系统内核以及 EOS 应用程序。	验证	2
2	**操作系统的启动**	● 跟踪调试 EOS 在 PC 机上从加电复位到成功启动的全过程，了解操作系统的启动过程。 ● 查看 EOS 启动后的状态和行为，理解操作系统启动后的工作方式。	验证	2
3	**进程的创建**	● 练习使用 EOS API 函数 CreateProcess 创建一个进程，掌握创建进程的方法，理解进程和程序的区别。 ● 调试跟踪 CreateProcess 函数的执行过程，了解进程的创建过程，理解进程是资源分配的单位。	验证	2
4	**线程的状态和转换**	● 调试线程在各种状态间的转换过程，熟悉线程的状态和转换。 ● 通过为线程增加挂起状态，加深对线程状态的理解。	验证 设计	2

5	**进程的同步**	● 使用 EOS 的信号量，编程解决生产者—消费者问题，理解进程同步的意义。 ● 调试跟踪 EOS 的信号量的工作过程，理解进程同步的原理。 ● 修改 EOS 的信号量算法，使之支持等待超时唤醒功能（有限等待），加深理解进程同步的原理。	验证设计	2
6	**时间片轮转调度**	● 调试 EOS 的线程调度程序，熟悉基于优先级的抢先式调度。 ● 为 EOS 添加时间片轮转调度，了解其它常用的调度算法。	验证设计	2
7	**物理存储器与进程逻辑地址空间的管理**	● 通过查看物理存储器的使用情况，并练习分配和回收物理内存，从而掌握物理存储器的管理方法。 ● 通过查看进程逻辑地址空间的使用情况，并练习分配和回收虚拟内存，从而掌握进程逻辑地址空间的管理方法。	验证设计	2
8	**分页存储器管理**	● 学习 i386 处理器的二级页表硬件机制，理解分页存储器管理原理。 ● 查看 EOS 应用程序进程和系统进程的二级页表映射信息，理解页目录和页表的管理方式。 ● 编程修改页目录和页表的映射关系，理解分页地址变换原理。	验证设计	2
9	**串口设备驱动程序**	● 调试 EOS 串口驱动程序向串口发送数据的功能，了解设备驱动程序的工作原理。 ● 为 EOS 串口驱动程序添加从串口接收数据的功能，进一步加深对设备驱动程序工作原理的理解。	验证设计	2
10	**磁盘调度算法**	● 通过学习 EOS 实现磁盘调度算法的机制，掌握磁盘调度算法执行的条件和时机。 ● 观察 EOS 实现的 FCFS、SSTF 和 SCAN 磁盘调度算法，了解常用的磁盘调度算法。 ● 编写 CSCAN 和 N-Step-SCAN 磁盘调度算法，加深对各种扫描算法的理解。	验证设计	2
11	**扫描 FAT12 文件系统管理的软盘**	● 通过查看 FAT12 文件系统的扫描数据，并调试扫描的过程，理解 FAT12 文件系统管理软盘的方式。 ● 通过改进 FAT12 文件系统的扫描功能，加深对 FAT12 文件系统的理解。	验证设计	2
12	**读文件和写文件**	● 了解在 EOS 应用程序中读文件和写文件的基本方法。 ● 通过为 FAT12 文件系统添加写文件功能，加深对 FAT12 文件系统和磁盘存储器管理原理的理解。	验证设计	2

表 1-2：配套实验清单

1.4 灵活选择实验内容

为了满足各种教学目标和教学方式，现有的配套实验具有如下特点：

- 覆盖了操作系统原理中所有重要的概念，包括进程管理、存储器管理、设备管理、文件系统等。

- 实验题目遵循由易到难的原则。前面的若干实验以较简单的验证性内容为主，后面的若干实验中加入了具有一定难度的设计性内容。
- 在单个实验中也遵循由易到难的原则。在每个实验的开始部分，都是较简单的验证性的内容，待学生在实验过程中对操作系统原理有一定理解后，再完成设计性的内容。有能力的学生还可以尝试完成实验后面的思考与练习。
- 除了第一个实验《实验环境的使用》应该首先完成外，其他实验几乎没有任何依赖关系。这样不但可以方便教师灵活选用实验题目，而且，学生在前面的实验没有完成的情况下，也可以顺利的进行后面的实验。

教师可以利用上述特点，根据不同的教学目标和教学方式，灵活选择实验题目并安排合理的实验顺序。例如：

教学类型	实验方式
高职、大专	教师可以选择一些相对较容易的实验，并且可以只安排学生完成验证部分的实验，对于有能力的学生可以选择完成设计部分的实验。
应用型本科	教师可以安排学生完成全部验证部分和设计部分的实验。
研究型本科	除了完成所有实验的内容外，教师还可以安排学生完成实验后面附加的思考与练习。

表 1-3：教学类型与实验方式

在教学时间充足的情况下，建议教师结合《EOS 操作系统实验教程》对 EOS 操作系统进行详细的讲解。如果教学时间只能够用来讲授操作系统原理教材中的内容，教师也可以在每个实验之前对涉及到的 EOS 操作系统相关知识进行简单的讲解，然后引导学生自学。具体的实验方式可以由教师灵活掌握。

1.5 应用到整个教学流程中

在将 OS Lab 应用于教学时，可以参照图 1-1 所示的典型流程来进行。

在学期开始前，教师可以首先根据教学安排，从已经提供的多个实验中选取若干个合适的实验供学生完成。为了取得较好的实验效果，建议教师可以按照下面的方式安排学生进行实验：

1. **实验开始前**。建议教师结合《EOS 操作系统实验教程》中的相关内容、本书第 2 章对应的内容、以及其它资料（包括配套的幻灯片、培训录像等）对实验进行适当的讲解，特别是实验中的难点和需要注意的地方。然后，安排学生对该实验进行必要的预习（例如阅读相关的文档，使用 OS Lab 演示版阅读与该实验相关的 EOS 源代码等）。
2. **实验进行中**。学生按照《EOS 操作系统实验教程》中的要求完成实验内容，并保存修改后的 EOS 源代码和操作信息（如图 1-2 所示）。
3. **实验结束后**。教师可以根据本书第 2 章中提供的检查实验结果的方法，检查学生完成实验的情况，最终达到使学生深入理解操作系统原理的目的。

配套的实验还可以用于课程设计。可以将授课过程中没有选用的实验题目留到课程设计中完成，如果授课过程中已经完成了所有实验，也可以将一些实验后面提供的问题或者练习留到课程设计中完成。

配套的实验题目以及实验后面提供的问题或者练习也可以做为毕业设计的题目，当然，

毕业设计的题目也可以是对 EOS 操作系统中某个模块的深入分析或者功能扩展，或者是在参考 EOS 的基础上重新编写一个小型的操作系统，具体的形式可以由教师和学生灵活掌握。

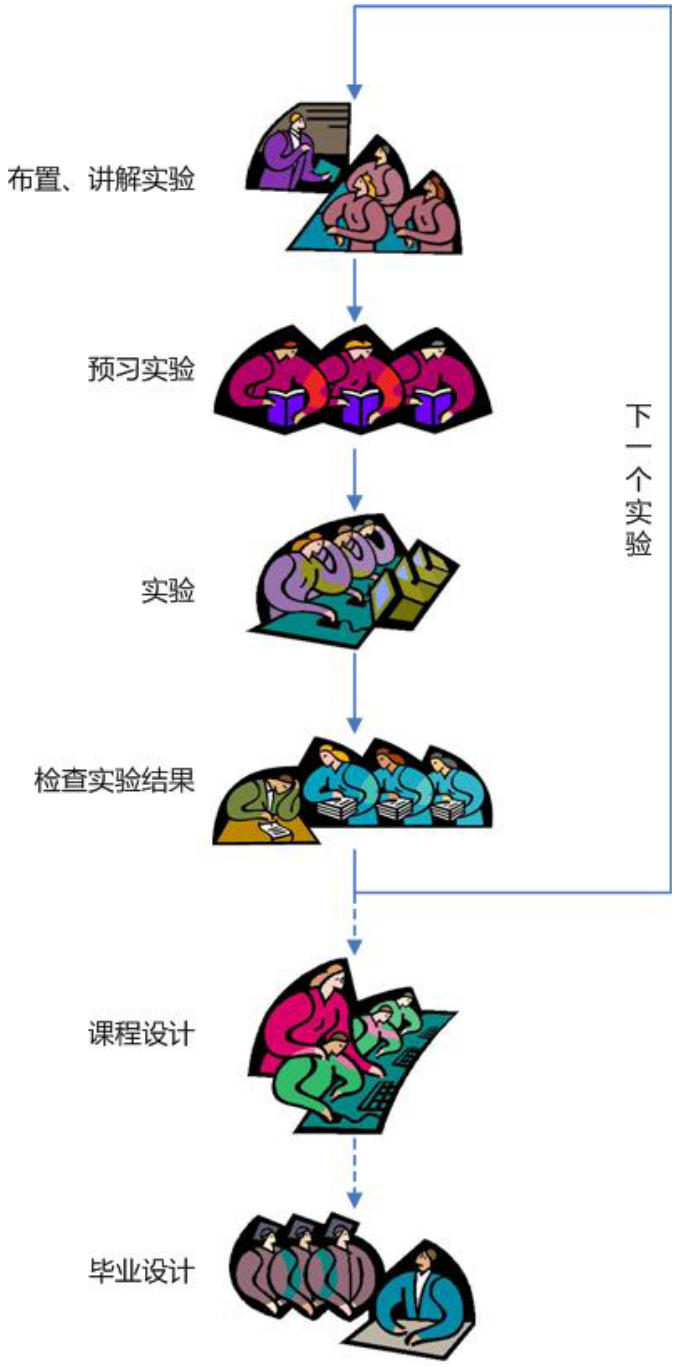

图 1-1：在整个教学流程中进行应用

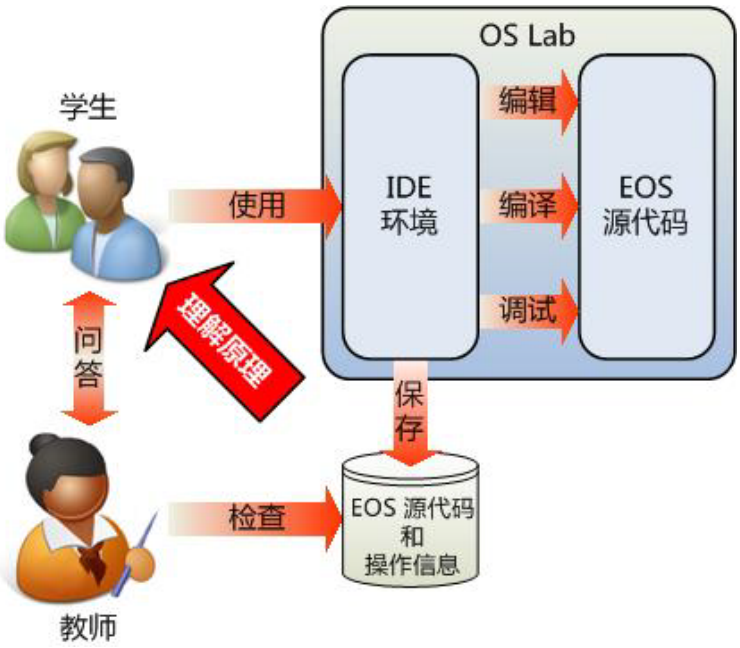

图 1-2：典型实验过程

1.6 检查实验结果

一般情况下，教师可以使用两种方式检查学生的实验结果

1. 登录到开放实验管理平台检查学生出勤情况、编写的代码量、提交的作业。
2. 让学生回答实验后面的问题。

其中，第一种方式最为简单、高效，不但可以准确判断学生是否自行完成实验，还可以大大减轻教师的负担。教师可在确认实验室内学生全部登录 OS Lab 后使用自动考勤功能记录学生的出勤情况。当学生自己编写代码后，开放实验平台会统计其编写的代码量。若学生是复制他人的代码，则代码量一项显示为 0，教师需要重点检查此类学生。当学生完成实验任务后，可将自己编写的代码提交到服务器端保存。待实验结束后，教师可以登录到开放实验管理平台查看学生编写的代码，更可将其下载到教师自己的机器上运行，从而轻松、准确地对每个学生的实验情况进行考评。教师可以在每完成一个实验之后就进行一次考评，也可以在期末进行一次考评。

注意，如果用于实验的计算机有硬盘保护功能，建议教师提醒学生将修改后的 EOS 源代码保存在没有被保护的磁盘上，或者保存在学生自带的移动存储器中，或者上传到服务器中，以防止实验数据丢失。

在本书第 2 章提供了实验问题的答案以及要求学生编写源代码的答案，但是这些仅做为参考，并不是唯一的答案。

1.7 技术支持与项目合作

教师在使用 OS Lab 的过程中，如果需要获得技术上的支持，可以随时与北京英真时代科技有限公司取得联系，获得专业的帮助。

如果现有的实验题目不能满足教学的需要，教师可以在现有实验题目的基础上自行设计新的实验题目，或者与北京英真时代科技有限公司合作设计新的实验题目。

总之，走“产学研”合作的道路，对于提高学校的教学质量和科研能力，以及提升企业的持续创新能力都具有十分重要的意义。

第 2 章 实验参考

本章中的每一个小节对应于一个实验。在每一个小节中提供了对应实验的答案和注意事项等内容，供教师参考。

2.1 实验环境的使用

2.1.1 准备实验

建议在实验前使用幻灯片向学生布置和讲解实验。

建议将产品光盘中的 OS Lab 演示版程序安装包分发给学生，这样学生就可以在自己的计算机上使用演示版程序预习和复习实验。

2.1.2 问题答案及参考代码

1. “逐过程”、“逐语句”和“跳出”

当在一个调用函数的代码行中断时，如果要查看被调用函数内部执行的具体情况，就需要使用“逐语句”功能，此功能会进入被调用函数并在被调用函数的第一条指令处中断执行。如果不关心此函数内部执行的具体情况，就需要使用“逐过程”功能，此功能会在被调用函数返回后的第一条指令处中断执行。如果使用“逐语句”功能调试进入被调用函数后，要立即跳出此函数继续调试（例如发现函数很复杂，只想立即查看函数执行的结果），就需要使用“跳出”功能，此功能也会在被调用函数返回后的第一条指令处中断执行。

2. EOS SDK 文件夹的目的和作用。

EOS SDK 文件夹主要是供 EOS 应用程序使用。EOS 内核提供的 API 函数及重要数据类型的定义都是通过将相关的头文件复制到 SDK 文件夹中，然后 EOS 应用程序再包含 SDK 文件夹中的这些头文件，使 EOS 应用程序能够调用 API 函数，或者使用已经定义好的数据类型来定义变量。同时，SDK 文件夹中还保存了 Debug 和 Release 版本的 EOS 的二进制文件，分别供 Debug 和 Release 版本的 EOS 应用程序使用。

3. EOS SDK 文件夹的结构及各个文件的来源和作用。

文件夹名			文件名	作用
SDK	bin	debug	boot.bin	软盘引导程序
			loader.bin	加载程序
			libkernel.a	导入库
			kernel.dll	内核（有调试信息）
		release	boot.bin	软盘引导程序
			loader.bin	加载程序
			libkernel.a	导入库
			kernel.dll	内核（无调试信息）
	inc		eos.h	导出 API 函数的声明。

		eosdef.h	导出数据类型的定义。
		error.h	导出错误码。

表 2-1：EOS SDK 文件夹的结构及各个文件的来源和作用

生成 Debug 配置的 EOS Kernel 项目后，将生成的二进制文件复制到 debug 文件夹中；生成 Release 配置的 EOS Kernel 项目后，将生成的二进制文件复制到 release 文件夹中。Inc 文件夹中的头文件是从 EOS Kernel 项目中复制到 SDK 文件夹中的。

libkernel.a 文件是导入库（import library）文件，与 Windows 动态链接库使用的 LIB 文件类似，用来导出 kernel.dll 中的导出函数（API 函数）。所以 EOS 应用程序在链接时需要使用此导入库文件来引用 EOS API 函数。

4. **EOS SDK 文件夹中头文件的包含关系。**

EOS 应用程序在项目的头文件中只是包含了 eos.h 文件，在 eos.h 文件中又包含了 eosdef.h 和 error.h 文件，所以这三个文件就都被包含了。

2.1.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生在实验过程中新建的项目（包括 Windows 控制台项目、EOS Kernel 项目和 EOS 应用程序项目）。

2.1.4 注意事项

- EOS 应用程序在启动调试之前必须删除所有的断点，否则启动调试会失败。如果在有断点的情况下启动调试了 EOS 应用程序，只需要结束此次调试，删除所有断点后再重新启动调试即可。建议向学生强调这一点，否则学生在调试 EOS 应用程序时可能会经常遇到类似的问题。
- 关于学生在关闭 OS Lab 后自动保存的 OUD 文件，建议教师安排学生将 OUD 文件保存到自带的 U 盘中，或者上传到服务器上统一管理。教师可以在每次实验结束后使用 TrackerWork 工具检查学生的实验情况，也可以在所有实验结束后再检查。

2.2 操作系统的启动

2.2.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容，可以重点讲解 EOS 操作系统的启动过程以及内存布局。

如果有必要，也可以让学生回忆一下之前学习的《IBM 汇编语言》和《计算机组成原理》中的相关知识，例如常用的汇编指令，实模式下的地址空间和分段内存管理，中断和中断向量表，以及 Intel X86 CPU 中主要寄存器（特别是 CS 和 IP 寄存器）的作用等。

2.2.2 问题答案及参考代码

1. **验证软盘引导扇区加载到内存后两个用户可用区域的高地址端是空白的调试命令**
   - 验证用户可用区域 1 的高地址端是空白的调试命令为 **xp /512b 0x7a00**。
   - 验证用户可用区域 2 的高地址端是空白的调试命令为 **xp /512b 0x9fe00**。
2. **验证上位内存的高地址端已经被系统占用的调试命令**
   xp /512b 0xffe00
3. **查看 Loader 程序结束位置字节码的调试命令**
   如果 loader.bin 文件的大小为 1566 个字节，转换为十六进制就是 61E，所以程序最后 8 个字节就在物理内存的 0x1616 到 0x161D 的位置，查看这 8 个字节的调试命令是 **xp /8b 0x1616**。当然，如果 loader.bin 文件的大小发生变化，命令也要随之变化。
4. **EOS 从软盘启动时要使用 boot.bin 和 loader.bin 两个程序的原因以及这两个程序各自的主要功能。**
   Boot.bin 是软盘引导扇区程序，大小只能是 512 个字节（与一个扇区的大小相同）。由于 EOS 从软盘启动的过程比较复杂，必然会造成引导程序较大，显然不能将引导程序全部放入软盘引导扇区中，这样就需要在 loader.bin 程序中完成部分功能。
   Boot.bin 程序的主要功能是将软盘根目录中的 loader.bin 文件加载到物理内存的 0x1000 处，然后跳转执行 loader.bin 程序的第一条指令，继续启动 EOS。Loader.bin 程序的主要功能是将软盘根目录中的 kernel.dll 文件加载到内存中，然后启动保护模式和分页机制，最后跳转到 kernel.dll 的入口点函数执行。
5. **软盘引导扇区加载完毕后内存中有两个用户可用的区域，为什么软盘引导扇区程序选择将 loader.bin 加载到第一个可用区域的 0x1000 处呢？这样做有什么好处？这样做会对 loader.bin 文件的大小有哪些限制？**
   在软盘引导扇区程序执行时，CPU 处于实模式状态，分段管理内存。由于此时代码段 CS 寄存器的值为 0（其它段寄存器的值也都为 0），所以在不改变段寄存器值的情况下，只能访问从 0x0000 到 0xFFFF 的地址空间。又因为软盘引导扇区程序受到 512 个字节的限制，所以应该尽量简单，当然就不希望出现修改段寄存器值的代码。所以最好能够将 loader.bin 文件加载到从 0x0000 到 0xFFFF 的地址空间中。现在有两个用户可用区域，在“用户可用 1”中能使用从 0x0600 到 0x7BFF 的 30208 个字节，在“用户可用 2”中能使用从 0x7E00 到 0xFFFF 的 33280 个字节，没有太大的区别，但是放入“用户可用 1”可以让更多“用户可用 2”的区域留给 kernel.dll 使用。同时，为了保持 4K 地址对齐，所以选择从 0x1000 处开始加载，这样在后面启动分页机制后，loader 程序

就是从一个页的 0 偏移处开始的。由于上面的原因，loader.bin 文件就只能使用从 0x1000 到 0x7BFF 的 27648 个字节，所以 loader.bin 文件的大小必须满足此限制。

2.2.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 向学生询问实验指导中提出的问题和学生设计的调试命令。

2.2.4 注意事项

- 实验调试的最后部分，在查看 kernel.dll 的入口点地址时，由于 kernel.dll 文件可能会发生变化，导致入口点地址变化，所以在实验指导中没有记录实际的入口点地址，但是这并不影响验证的过程。
- 实践证明，有相当一部分学生不知道如何使用 Windows 操作系统正确的查看一个文件的大小（不是文件占用的磁盘空间）。建议教师对这个概念进行适当的介绍，避免学生将此概念混淆。例如，要查看 loader.bin 文件的大小，应该在 Windows 资源管理器中右键点击此文件，选择快捷菜单中的“属性”，弹出的“属性”对话框如图 2-1 所示：

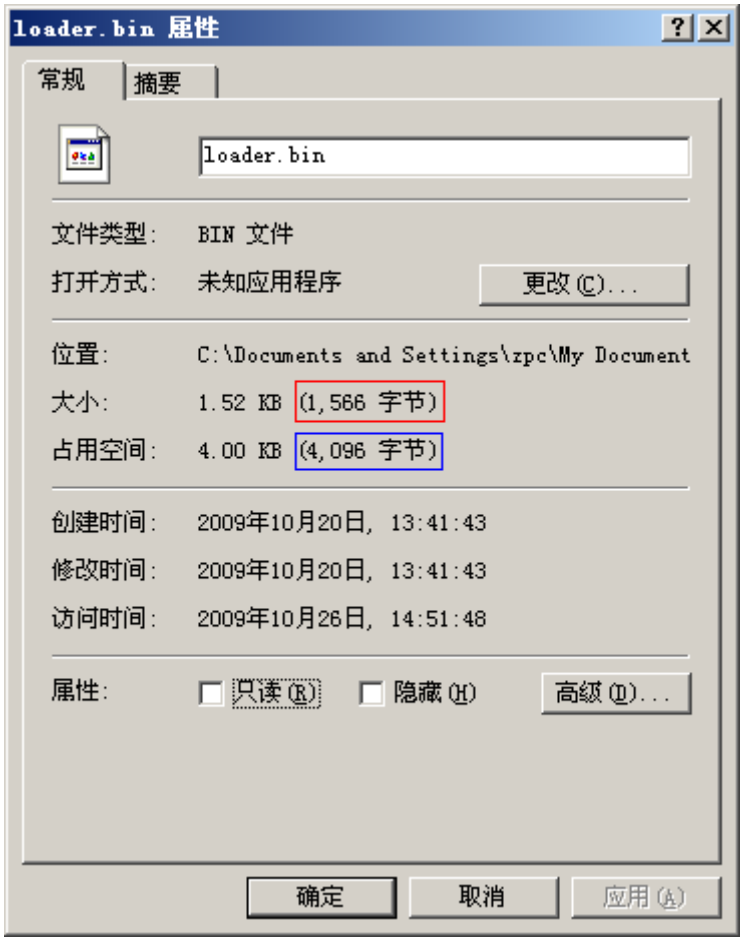

图 2-1：在“属性”对话框中查看文件的大小和占用空间。

红色框中的数据才是 loader.bin 文件的大小，是字节的整数倍。而蓝色框中的数据是 loader.bin 文件在磁盘上占用的空间大小，是簇的整数倍（这里一个簇是 4096 字节）。

2.3 进程的创建

2.3.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。讲解进程控制块 PCB 的结构和作用；讲解如何通过编程的方式让一个应用程序创建另外一个应用程序的进程；讲解与创建进程相关的各个 EOS API 函数的用法。

在实验前需要将学生包中的源代码文件分发给学生。如果学生编程能力有限，可以结合流程图或者直接使用参考源代码讲解这些源代码。

在讲解 NewProc.c 文件 main 函数中的源代码时建议注意下面几个方面：

- CreateProcess 函数中前四个参数是输入参数，最后一个参数是输出参数。
- CreateProcess 函数的后两个参数虽然是两个结构体变量的指针，但是不能定义这两个结构体的指针变量（未分配内存）来做为参数，而必须定义这两个结构体的变量（分配了内存），然后将变量的指针做为参数。
- CreateProcess 函数的第一个参数是一个字符串常量，注意字符串常量中的反斜线”\”必须写成双反斜线”\\”进行转义。

2.3.2 问题答案及参考代码

1. 进程创建过程的流程图

在图 2-2 中只提供了一个较简单的流程图，还可以加入分支来表示错误处理。

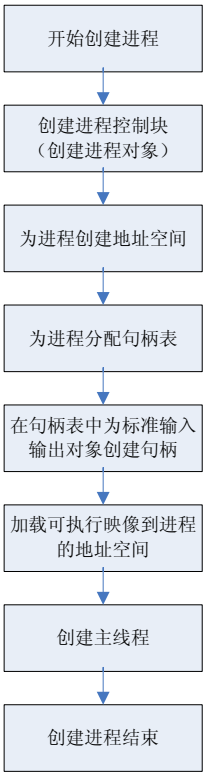

图 2-2：进程创建过程的流程图。

2. **在 PsCreateProcess 函数中调用了 PspCreateProcessEnvironment 函数后又先后调用了 PspLoadProcessImage 和 PspCreateThread 函数，这些函数的主要功能是什么？能够交换这些函数被调用的顺序吗？原因是什么？**

   PspCreateProcessEnvironment 的主要功能是创建进程控制块，并且为进程创建了地址空间和分配了句柄表。PspLoadProcessImage 是将进程的可执行映像加载到了进程的地址空间中。PspCreateThread 创建了进程的主线程。这三个函数被调用的顺序是不能够改变的，就向上面描述的，加载可执行映像之前必须已经为进程创建了地址空间，这样才能够确定可执行映像可以被加载到内存的什么位置;在创建主线程之前必须已经加载了可执行映像，这样主线程才能够知道自己要从哪里开始执行，执行哪些指令。

3. **Hello.exe 应用程序的源代码**

   请参考本实验文件夹中的 Hello.c 源文件。

4. **让应用程序创建另一个应用程序的进程的源代码**

   请参考本实验文件夹中的 NewProc.c 源文件。

5. **为应用程序同时创建多个进程的源代码**

   请参考本实验文件夹中的 NewTwoProc.c 源文件。

2.3.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 向学生询问实验指导中提出的问题，检查学生绘制的进程创建过程的流程图。
- 登录到开放实验管理平台下载学生编写的代码，执行代码看是否能够正常创建进程。

2.3.4 调整难度

如果学生的编程能力比较强，教师可以适当增加本实验的内容或难度。例如可以考虑让学生在学习了 NewProc.c 文件中的源代码后，根据要求自己编写同时创建多个进程的源代码，而不是直接将 NewTwoProc.c 源文件提供给学生。

当然，如果学生的能力非常强，也可以考虑让学生根据 main 函数的流程图来编写创建进程的源代码，而不将 NewProc.c 源文件提供给学生。

2.4 线程的状态和转换

2.4.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解线程的状态和转换过程，以及 Suspend 和 Resume 原语。

2.4.2 问题答案及参考代码

1. **修改后的 PsSuspendThread 函数。**
   请参见教师包本实验文件夹中的 resume.c 文件。
2. **当 loop 线程处于运行状态时，EOS 中还有哪些线程，它们分别处于什么状态。**
   当 loop 线程处于运行状态时，EOS 中的线程有：
   - 一个优先级为 0 的空闲线程。处于就绪状态。
   - 4 个优先级为 24 的控制台线程。处于阻塞状态。
   - 1 个优先级为 24 的控制台派遣线程。处于阻塞状态。
3. **当 loop 线程在控制台 1 中执行，并且在控制台 2 中执行 suspend 命令时，为什么控制台 1 中的 loop 线程处于就绪状态而不是运行状态？**
   当在控制台 2 中执行 suspend 命令时，实质上是优先级为 24 的控制台 2 线程抢占了处理器，也就是控制台 2 线程处于运行状态，所以此时 loop 线程就处于就绪状态了。
4. **总结一下，哪些线程状态转换过程需要使用线程控制块中的上下文（将线程控制块中的上下文恢复到处理器中，或者将处理器的状态复制到线程控制块的上下文中），哪些不需要使用，并说明原因。**
   - 将新建线程转入就绪状态时，需要初始化线程控制块中上下文（在文件 ps/i386/pscxt.c 中的 PspInitializeThreadContext 函数中完成）。
   - 线程由就绪状态进入运行状态时，需要将线程控制块中的上下文恢复到处理器中，这是在中断返回前完成的。
   - 线程由运行状态进入就绪状态或阻塞状态时，需要将处理器的状态复制到线程控制块的上下文中，这是在进入中断时完成的。
   - 线程由阻塞状态进入就绪状态时，不需要使用线程控制块中的上下文。
   - 线程由任意状态进入结束状态时，如果是由运行状态被结束，则还需要在中断时保存上下文（不可避免），然后再结束。如果是其它状态被结束，就不需要使用线程控制块中的上下文。

2.4.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。。
- 检查学生编写的 Resume 原语。可以查看学生编写的源代码，也可以使用学生修改后的 resume 命令，查看是否能让挂起的线程恢复执行。

2.4.4 调整难度

该实验要求学生完成的代码相对比较简单，并且已经提供了大量的提示。如果学生完成仍然比较困难，建议教师在实验前可以对源代码参考答案进行适当讲解，帮助学生顺利完成实验。

2.5 进程的同步

2.5.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解生产者－消费者的同步问题以及 EOS 用来进行线程同步的内核对象（Mutex 和 Semaphore）。Event 对象在本实验中没有涉及到，可以让学生自学。EOS 应用程序创建线程的方法不是重点，也可以让学生自学。

在开始实验前建议将学生包中的两个源文件分发给学生。

建议结合流程图对 pc.c 文件中的源代码进行适当的讲解。在讲解时可以重点介绍下面几个方面：

1. 在 pc.c 文件中定义了一个有 10 个缓冲区的缓冲池，每个缓冲区中可以放入一个整型数据（一个整型数据代表一个产品），同时定义了产品数量为 30。
2. 在 main 函数中先后创建了三个 EOS 内核同步对象，分别是：用于互斥访问缓冲池的 Mutex（MutexHandle）；Empty 信号量（EmptySemaphoreHandle），表示缓冲池中空缓冲区的数量；Full 信号量（FullSemaphoreHandle），表示缓冲池中满缓冲区的数量。
3. 在创建两个信号量时向 CreateSemaphore 函数传递的参数是不同的。
4. 这里使用的是线程而不是进程，但是同步的概念是一致的。
5. 在生产者线程和消费者线程中共享的数据要定义为全局的。例如，缓冲池和同步对象。

2.5.2 问题答案及参考代码

1. 生产者线程函数和消费者线程函数中的临界资源是什么？

目前只有缓冲池是临界资源。

2. 生产者线程函数和消费者线程函数中分别有哪些代码是临界区？

包含在进入临界区代码和退去临界区代码之间的代码就是临界区。生产者线程函数的临界区为

```
printf("Produce a %d\n", i);
Buffer[InIndex] = i;
InIndex = (InIndex + 1) % BUFFER_SIZE;
```

消费者线程函数的临界区为

```
printf("\t\t\tConsume a %d\n", Buffer[OutIndex]);
OutIndex = (OutIndex + 1) % BUFFER_SIZE;
```

3. 生产者线程函数和消费者线程函数中进入临界区和退出临界区的代码是哪些？是否成对出现？

进入临界区的代码为

```
WaitForSingleObject(MutexHandle, INFINITE);
```

退出临界区的代码为

```
ReleaseMutex(MutexHandle);
```

进入临界区和退出临界区的代码必须成对出现。

4. **生产者线程和消费者线程是如何使用 Mutex、Empty 信号量和 Full 信号量来实现同步的？这两个线程函数中对这三个同步对象的操作能够改变顺序吗？**

Mutex 对象只是用来保护临界资源的，防止生产者线程和消费者线程同时访问临界资源，从而造成混乱。Empty 信号量和 Full 信号量是相互合作不可分割的，在生产者线程中，Empty 信号量保证在没有空缓冲区的时候让生产者停止生产，Full 信号量保证在生产完毕一个产品后增加一个满缓冲区；在消费者线程中，Full 信号量保证在没有满缓冲区的时候让消费者停止消费，Empty 信号量保证在消费完毕一个产品后增加一个空缓冲区。所以对这三个同步对象的操作是不能改变顺序的。

5. **为信号量添加等待超时唤醒功能和批量释放功能后的参考代码**

下面列出的代码仅仅是实现所要功能的一种方法，完全可以使用不同的代码来实现相同的功能。红色的是发生了变化的代码。

EOS Kernel 项目中 ps/semaphore.c 文件的 PsWaitForSemaphore 函数应该被修改为

```
STATUS
PsWaitForSemaphore(
    IN PSEMAPHORE Semaphore,
    IN ULONG Milliseconds
    )
{
    STATUS Status;
    BOOL IntState;

    ASSERT(KeGetIntNesting() == 0); // 中断环境下不能调用此函数。

    IntState = KeEnableInterrupts(FALSE); // 开始原子操作，禁止中断。

    if(Semaphore->Count > 0) {
        Semaphore->Count--;
        Status = STATUS_SUCCESS;
    } else {
        Status = PspWait(&Semaphore->WaitListHead, Milliseconds);
    }

    KeEnableInterrupts(IntState); // 原子操作完成，恢复中断。

    return Status;
}
```

EOS Kernel 项目中 ps/semaphore.c 文件的 PsReleaseSemaphore 函数应该被修改为

```
STATUS
PsReleaseSemaphore(
    IN PSEMAPHORE Semaphore,
    IN LONG ReleaseCount,
    OUT PLONG PreviousCount
    )
{
```

```
    STATUS Status;
    BOOL IntState;

    IntState = KeEnableInterrupts(FALSE); // 开始原子操作，禁止中断。

    if (Semaphore->Count + ReleaseCount > Semaphore->MaximumCount) {

        Status = STATUS_SEMAPHORE_LIMIT_EXCEEDED;

    } else {

        //
        // 记录当前的信号量的值。
        //
        if (NULL != PreviousCount) {
            *PreviousCount = Semaphore->Count;
        }

        //
        // 释放信号量。
        //
        for (Semaphore->Count += ReleaseCount;
            Semaphore->Count > 0 && !ListIsEmpty(&Semaphore->WaitListHead);
            Semaphore->Count--) {
            // 被唤醒线程从 PsWait 返回时的返回值为 STATUS_SUCCESS。
            PspWakeThread(&Semaphore->WaitListHead, STATUS_SUCCESS);
        }

        //
        // 可能有线程被唤醒进入就绪状态，需要执行调度。
        //
        PspThreadSchedule();

        Status = STATUS_SUCCESS;
    }

    KeEnableInterrupts(IntState); // 原子操作完成，恢复中断。

    return Status;
}
```

2.5.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。。

- 向学生询问实验指导中提出的问题。
- 学生在添加信号量超时等待功能后，检查学生修改的 EOS Kernel 项目和生产者－消费者项目中的代码。执行学生修改后的生产者－消费者项目，看生产者线程和消费者线程是否能够同步执行，可以将输出的结果与标准结果（使用本参考提供的代码实现的项目所输出的结果）进行比较。

2.5.4 调整难度

如果学生在修改信号量算法时困难较大，建议教师可以考虑将参考代码提供给学生。

可以考虑让学生实现多个生产者线程和消费者线程的同步执行，此时同步问题会复杂很多，让学生加深对进程同步的理解。

2.6 时间片轮转调度

2.6.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 EOS 目前使用的基于优先级的抢先式调度，调度程序执行的时机和过程，以及如何实现时间片轮转调度。

注意，不要将教师包中的 rr.c 文件分发给学生，本实验要求学生自己编写 rr.c 文件中的源代码。

2.6.2 问题答案及参考代码

1. 时间片轮转调度的参考代码

请参见教师包中的 rr.c 文件。

2. 使用“关中断”和“开中断”保护临界资源的原因

关中断后 CPU 就不会响应任何由外部设备发出的硬中断（包括定时计数器中断和键盘中断等）了，也就不会发生线程调度了，从而保证各个线程可以互斥的访问控制台。

这里绝对不能使用互斥信号量（MUTEX）保护临界资源的原因：如果使用互斥信号量，则那些由于访问临界区而被阻塞的线程，就会被放入互斥信号量的等待队列，就不会在相应优先级的就绪队列中了，而时间片轮转调度算法是对就绪队列的线程进行轮转调度，而不是对这些被阻塞的线程进行调度，也就无法进行实验了。使用“关中断”和“开中断”进行同步就不会改变线程的状态，可以保证那些没有获得处理器的线程都在处于就绪队列中。

3. 使低优先级线程也能获得执行机会的调度算法

在 ke/sysproc.c 文件中的 ConsoleCmdRoundRobin 函数调用 Sleep 函数语句的后面添加下面的语句，即可以演示高优先级线程抢占处理器后，低优先级线程无法运行的情况，待高优先级线程结束后，低优先级线程才能够继续运行。

```
    HANDLE ThreadHandle;
    THREAD_PARAMETER ThreadParameter;
    __asm("cli");
    ThreadParameter.Y = 20;
    ThreadParameter.StdHandle = StdHandle;
    ThreadHandle = (HANDLE)CreateThread(
        0, ThreadFunction, (PVOID)&ThreadParameter, 0, NULL);
    PsSetThreadPriority(ThreadHandle, 9);
    __asm("sti");
    Sleep(10 * 1000);
    TerminateThread(ThreadHandle, 0);
    CloseHandle(ThreadHandle);
    Sleep(10 * 1000);
```

解决该问题的最简单的方法是实现动态优先级算法。动态优先级是指在创建线程时所赋予的优先级，可以随线程的推进而改变，以便获得更好的调度性能。例如，可用规定，在就绪队列中的线程，随着其等待时间的增长，其优先级以速率 x 增加，并且正在执行的线程，

其优先级以速率 y 下降。这样，在各个线程具有不同优先级的情况下，对于优先级低的线程，在等待足够的时间后，其优先级便可能升为最高，从而获得被执行的机会。此时，在基于优先级的抢先式调度算法、时间片轮转调度算法和动态优先级算法的共同作用下，可防止一个高优先级的长作业长期的垄断处理器。

当然，还有很多其它的调度算法可以解决类似的问题。

4. Windows、Linux 中的时间片轮转调度

在 Windows、Linux 等操作系统中，虽然都提供了时间片轮转调度算法，但时间片轮转调度算法却很少真正被派上用场，下面解释原因。

在 Windows 中打开任务管理器，如图 2-3：

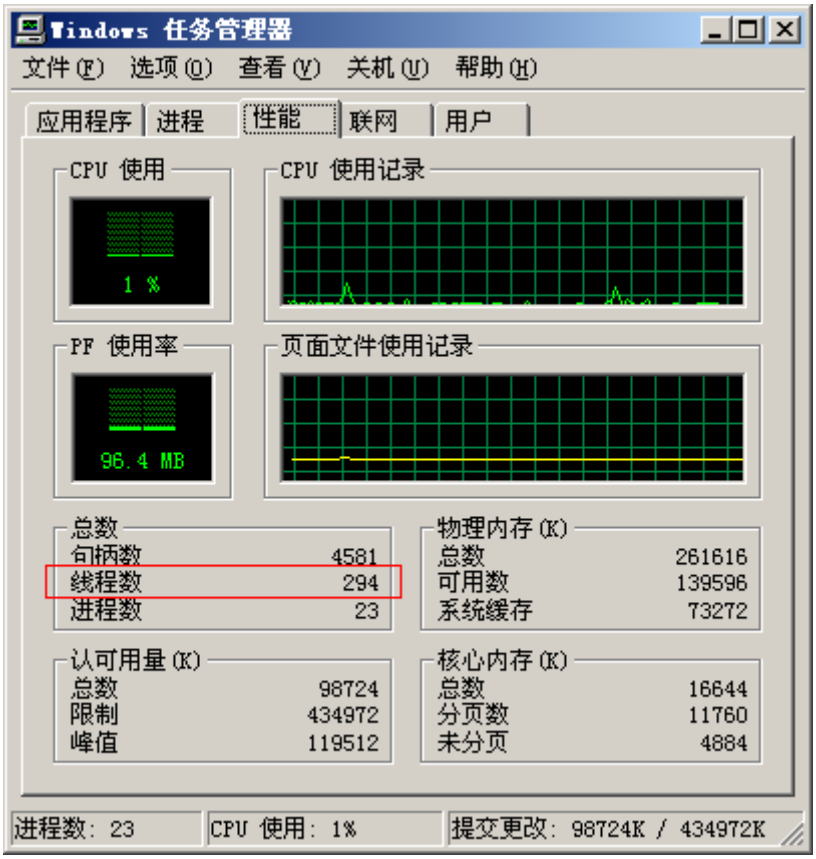

图 2-3：Windows 任务管理器。

可以发现，即便系统中已经运行了数百个线程，但 CPU 的利用率仍然很低，甚至为 0。为什么会这样呢？因为这些线程在大部分时间都处于阻塞状态，阻塞的原因是各种各样的，最主要的原因是等待 I/O 完成或者等待命令消息的到达。例如，在编辑 Word 文档时，每敲击一次键盘，Word 就会立即作出反应，并在文档中插入字符。此时，会感觉 Word 运行的非常流畅。事实上，并非如此，Word 主线程大部分时间都处于阻塞等待状态，等待用户敲击键盘。在用户没有敲击键盘或没有使用鼠标点击时，Word 主线程处于阻塞状态，它将让出处理器给其它需要的线程。当用户敲击一个按键后，Word 主线程将会立刻被操作系统唤醒，此时 Word 开始处理输入请求。Word 在处理输入请求时所用的 CPU 时间是非常短的（因为 CPU 非常快），是微秒级的，远远低于时间片轮转调度的时间片大小（Windows 下是 60 毫秒），处理完毕后 Word 又立刻进入阻塞状态，等待用户下一次敲击键盘。

或者拿音乐播放器来分析，表面上感觉播放器在不停播放音乐，但是 CPU 的利用率仍然会很低。这是由于播放器会将一段声音编码交给声卡，由声卡来播放，在声卡播放完这段声音之前，播放器都是处于阻塞等待状态的。当声卡播放完片段后，播放器将被唤醒，然后

它会将下一个声音片段交给声卡继续播放。

掌握了上面的知识后，就可以很容易解释为什么这么多线程同时在运行而一点都感觉不到轮替现象。线程大部分时间都是处于阻塞状态，当某个线程阻塞等待的事件到达后，线程被系统唤醒，因为其它线程大都处于阻塞状态，CPU 处于空闲状态的几率非常高，所以线程被唤醒后几乎就可以立即获得 CPU 运行。线程在很短时间内运行处理完事件，然后重新阻塞并释放 CPU，而不是一直占用 CPU 直到时间片用完。

2.6.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的时间片轮转调度。可以查看学生编写的源代码，也可以使用学生修改后的 EOS 内核执行控制台命令“rr”，查看是否所有的线程能够轮流执行。

2.6.4 调整难度

该实验要求学生完成的代码相对比较简单，并且已经在《实验指导》中提供了大量的提示和流程图。如果学生完成仍然比较困难，建议教师在实验前可以对源代码参考答案进行适当讲解，帮助学生顺利完成实验。

2.7 物理存储器与进程逻辑地址空间的管理

2.7.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 EOS 物理存储器的管理方式和进程逻辑地址空间的管理方式。

注意，不要将教师包中的 LoopAppEx.c 文件分发给学生，本实验要求学生自己编写 LoopAppEx.c 文件中的源代码。

2.7.2 问题答案及参考代码

1. **在应用程序进程中分配虚拟页和释放虚拟页的源代码。**
   请参见教师包中的 LoopAppEx.c 文件。
2. **如果分配了物理页后，没有回收，会对EOS操作系统造成什么样的影响？**
   EOS操作系统将不能再使用未回收的物理页，如果分配的物理页都没有进行回收，可能会造成EOS没有可用的物理页，从而导致EOS停止运行。目前EOS操作系统内核函数MiAllocateAnyPages还没有处理没有物理页可分配的情况。
3. **尝试从性能的角度分析内核函数MiAllocateAnyPages和MiAllocateZeroedPages。尝试从安全性的角度分析分配零页的必要性。**
   从性能的角度来分析，调用MiAllocateAnyPages函数分配物理页在某些情况下比调用MiAllocateZeroedPages函数要快速。
   从安全性的角度来分析，分配零页更加安全。例如，一个物理页被操作系统存储过重要的密码信息后被释放，如果没有清零就被分配给用户程序，则用户程序就可能从这个物理页中获取重要的密码信息。
4. **虚拟页描述符链表中产生空隙的原因。**
   产生的空隙是由于有虚拟页被释放造成的。在EOS启动时有一个初始化线程在初始化完毕后就退出了，线程的堆栈所占用的虚拟页就被释放了。
5. **MEM_RESERVE标志和MEM_COMMIT标志的区别。**
   使用MEM_RESERVE标志分配虚拟页时，没有为其映射实际的物理页。使用MEM_COMMIT表示分配虚拟页时，会为其映射实际的物理页。

2.7.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的在应用程序进程中分配虚拟页和释放虚拟页的源代码。可以查看学生编写的源代码，也可以使用学生编写的源代码生成 EOS 应用程序，并查看其执行的效果。

2.8 分页存储器管理

2.8.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解：

1. 讲解 C 语言中的位域结构体概念，可以帮助学生理解 PTE 结构体的定义。
2. i386 处理器的二级页表硬件机制，及逻辑地址到物理地址的转换过程。
3. EOS 的页目录和页表在进程 4G 虚拟地址空间中的位置。当 EOS 操作系统启动 i386 处理器的分页机制后，就只能使用逻辑地址（虚拟地址）来访问物理内存，所以只有知道了页目录和页表的虚拟地址，才能够访问它们。

   在 memory.c 文件中就是使用这些虚拟地址来访问页目录和页表的。虽然打印出了页目录和页表映射的物理页框号，但是这仅仅是为了演示它们的映射关系，是没有办法直接访问物理内存的。
4. 页目录在虚拟地址空间中的基址的获得方式：页目录的第 0x300 个 PDE 映射的页表就是页目录本身，这样页表（页目录）的第 0x300 个 PTE 又映射到页目录，所以用 PDE 标号（0x300）做 32 位线性地址的高 10 位，用 PTE 标号（0x300）做 12－22 位，就得到了地址 0xC0300000。

   同理，页表在虚拟地址空间中基址的获得方式：页目录的第 0x300 个 PDE 映射的页表是页目录本身，而且页目录的第 0x0 个 PDE 映射了第 0x0 个页表，所以页表（页目录）的第 0x0 个 PTE 就映射到第 0x0 个页表。用 PDE 标号（0x300）做 32 位线性地址的高 10 位，用 PTE 标号（0x0）做 12－22 位，就得到了地址 0xC0000000。
5. 在 memory.c 的 main 函数中是一个二重循环。
6. 在 memory.c 的 main 函数中使用输出函数 printf 将内容输出到标准输出（屏幕）。
7. Windows 科学计算器可以帮助学生进行十六进制、十进制和二进制计算，对于计算线性地址对应的物理地址很重要，建议教师对该工具的用法进行适当讲解。在 Windows“开始”菜单“程序”中选择“附件”，即可找到“计算器”工具。在计算器工具的“查看”菜单中选择“科学型”，即可进行多种进制数值的计算和转换。

2.8.2 问题答案及参考代码

1. 图 16-2（a）中应用程序进程的页目录和页表占用了几个物理页？页框号分别是多少？

页目录占用一个物理页，页框号是 0x409。页表占用 5 个物理页（不算复用为页表的页目录），页框号是 0x41D、0x401、0x403、0x404、0x402。

2. 图 16-2（a）中应用程序进程映射用户地址空间（低 2G）的页表的页框号是多少？该页表有几个有效的 PTE，或者说有几个物理页用来装载应用程序的代码、数据和堆栈，页框号是多少？

映射用户地址空间的页表的页框号是 0x41D。该页表有 11 个有效的 PTE，页框号是 0x41E，0x41F，0x420，0x421，0x422，0x423，0x424，0x425，0x426，0x427，0x428。

3. 比较图 16-2（a）和（b），系统进程和应用程序进程是否有各自的页目录？在页目录映射的页表中，哪些是独占的，哪些是共享的？

系统进程和应用程序进程一定有各自的页目录。映射了用户地址空间的页表被应用程序进程独占，页框号是 0x41D，其它的页表（映射了内核地址空间）都是共享的。

4. **统计当应用程序进程和系统进程并发时，总共有多少物理页被占用？**

   在图 16-2 中，以应用程序进程占用的物理页为基准，系统进程占用的物理页只有页目录与其不同，所以应用程序进程占用的物理页 1066 加上系统进程页目录占用的物理页 1 就是 1067。

5. **应用程序进程结束后，为什么系统进程（即内核地址空间）占用的物理页会减少？**

   应用程序结束后，EOS 内核会删除应用程序进程在内核地址空间中占用的内存（例如删除 PCB 对象等）。这些内存必须要回收，否则如果一个应用程序反复运行多次，内核空间就有被耗尽的危险，操作系统的可靠性就无从谈起了。

6. **两个应用程序进程并发时，进程 2 共享进程 1 代码页的映射方式。**

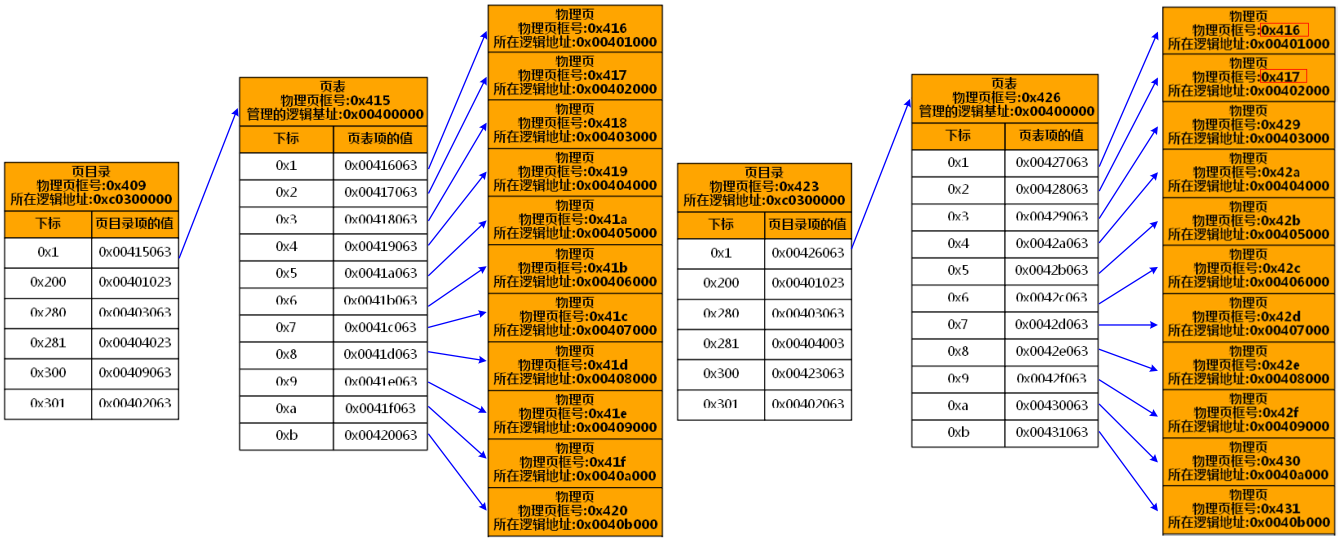

图 2-4：两个应用程序进程并发时，进程 2 共享进程 1 代码页的映射方式。

7. **统计当两个应用程序进程并发时，总共有多少物理页被占用？有更多的进程同时运行呢？如果进程的数量足够多，物理内存是否会用尽，如何解决该问题？**

   在图 16-3 中，以进程 1 占用的物理页为基准，进程 2 共有 14 个物理页与其不同，所以 1069＋14＝1083。如果有更多的进程同时运行，就会有更多的物理页被占用，由于物理页的数量是有限的，所以肯定会被用尽。使用虚拟内存技术（即将一些暂时不用的物理页存储在磁盘上，需要使用时再放入物理内存），可以解决该问题。

8. **假设修改了页目录，使其第 0x100 个 PDE 映射的页框号是页目录本身，此时页目录和页表会映射在 4G 虚拟地址空间的什么位置呢？说明计算方法。**

   - 页目录：PDE 标号 0x100 做为虚拟地址的高 10 位，PTE 标号 0x100 做为虚拟地址的 12-22 位，得到虚拟地址 0x 40100000。
   - 页表：PDE 标号 0x100 做为虚拟地址的高 10 位，PTE 标号 0x0 做为虚拟地址的 12-22 位，得到虚拟地址 0x 40000000。

9. **通过编程的方式统计并输出用户地址空间占用的内存数目。通过编程的方式统计并输出页目录和页表的数目。打印输出引导扇区所在的虚拟内存。**

   参见源代码文件 memoryex.c。

10. **为什么不能从页表基址 0xC0000000 开始遍历，来查找有效的页表呢？**

    只有当一个虚拟地址通过二级页表映射关系能够映射到实际的物理地址时，该虚拟地址才能够被访问，否则会触发异常。由于并不是所有的页表都有效，所以不能从页表基址 0xC0000000 开始遍历。

11. 编写代码将申请到的物理页从二级页表映射中移除，并让内核回收这些物理页。

参见源代码文件 MapNewPageEx.c。使用该文件中的 ConsoleCmdMemoryMap 函数替换 ke/sysproc.c 中的 ConsoleCmdMemoryMap 函数即可。

提示：在移除映射的物理页时，只需要将 PTE/PDE 的存在标志位设置为 0 即可，要先修改 PTE，再修改 PDE。另外，要注意刷新快表。调用 MiFreePages 函数即可回收物理页，具体的用法可以参考其函数定义处的注释和源代码(mm/pfnlist.c 第 248 行)。

2.8.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生从 OS Lab“输出”窗口保存的输出结果。
- 检查学生编写的 EOS 内核项目和 EOS 应用程序项目。

2.8.4 调整难度

该实验主要是观察验证的实验，比较简单。可以鼓励有能力的学生尽量完成实验后面的思考题。

2.9 页面置换算法与动态内存分配

2.9.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 OPT、FIFO、LRU、LFU、Clock 等页面替换算法；边界标识法和伙伴系统的内存分配和回收方法。

2.9.2 问题答案及参考代码

1. **实验第 3.1.3 小节，最近最久未使用页面置换算法。**

   请参见教师包本实验对应文件夹下的 page2.c 文件。

2. **实验第 3.2.3 小节，伙伴系统的设计。**

   请参见教师包本实验对应文件夹下的 buddy.c 文件。

3. **新建一个 EOS kernel 的项目，打开 mm 模块下的 virtual.c 文件和 syspool.c 文件，仔细阅读其中的源代码及注释，查看一下 EOS 系统动态分配和回收内存的方法。系统使用的方法是与本实验使用的边界标识法和伙伴系统法不同。请读者尝试分别用边界标识法和伙伴系统算法修改系统的动态分配内存和回收内存的方法。**

   略。开放性练习，需读者自己来完成。

4. **x86 平台为实现改进型的 Clock 页面置换算法提供了硬件支持，它实现了二级映射机制，它定义的页表项的格式如下：**

图9-1 页目录和页表中的表项的格式

其中位 5 是访问标志，当处理器访问页表项映射的页面时，页表表项的这个标志就会被置为 1。位 6 是页面修改标志，当处理器对一个页面执行写操作时，该项就会被置为 1。

在 EOS 的内核中写一个系统调用函数来实现改进型的 Clock 页面置换算法的模拟。

略。开放性练习，需读者自己来完成。

2.9.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的页面置换算法和伙伴系统。可以查看学生编写的源代码，也可以运行学生修改后的可执行文件，查看是否能够正常演示页面替换算法和伙伴系统。

2.10 串口设备驱动程序

2.10.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 EOS 串口驱动程序的 SrlWrite 和 SrlIsr 函数源代码。

在开始实验前应该将学生包中的 Serial.c 和 Serial.exe 分发给学生，Serial.c 文件中保存了 Serial.exe 应用程序的源代码。注意，不要将教师包中的“serial 参考答案.c”文件分发给学生，该实验要求学生自己编写此文件中的源代码。

2.10.2 问题答案及参考代码

1. 缓冲区对串口驱动程序执行效率造成的影响

在向串口发送数据时可以不使用缓冲区，将 SrlWrite 函数体修改为

```
{
    CHAR Data;
    ULONG Count;
    PDEVICE_EXTENSION Ext =
        (PDEVICE_EXTENSION)DeviceObject->DeviceExtension;

    PsResetEvent(&Ext->CompletionEvent);

    for (Count = 0; Count < Request; Count++) {

        Data = ((PCHAR)Buffer)[Count];

        WRITE_PORT_UCHAR(REG_PORT(DeviceObject, THR), Data);

        PsWaitForEvent(&Ext->CompletionEvent, INFINITE);
    }

    *Result = Count;
    return STATUS_SUCCESS;
}
```

将 SrlIsr 函数体修改为

```
{
    CHAR Data;
    PDEVICE_EXTENSION Ext =
        (PDEVICE_EXTENSION)Device->DeviceExtension;

    if (2 == READ_PORT_UCHAR(REG_PORT(Device, IIR))) {
```

```
            PsSetEvent(&Ext->CompletionEvent);

        } else  {

            // 省略

        }
}
```

这样 SrlWrite 函数每次向 THR 寄存器写入一个字符，然后阻塞，当 THR 寄存器中的数据被发送出去，由中断处理程序唤醒后继续发送下一个字符。

可以看到，在不使用缓冲区的情况下，每发送一个数据都要执行阻塞、唤醒操作，而且唤醒操作只是将线程转换为“就绪”状态，只有在发生线程调度时，线程才有可能继续发送数据，所以，在使用缓冲区的情况下可以大大减少线程调度等操作，可以提高 CPU 利用率。

缓冲区的大小对程序执行的效率也有很大影响。例如缓冲区大小是 8，而要发送 16 个字节的数据，此时，发送线程在向缓冲区放入 8 个字节数据后就会阻塞，当中断处理程序将缓冲区中的数据发送完毕后会唤醒发送线程，从而继续发送剩余的 8 个字节数据，而如果缓冲区大小是 256，一次就可以将 16 个字节的数据发送完毕，同样可以减少线程调度等操作。但是缓冲区也不是越大越好，如果缓冲区占用的内存空间很大，而串口设备很少被使用，就会造成内存的浪费。应该根据系统的特点以及应用的场所来设置合理的缓冲区大小。

2. **在 SrlRead 函数中，访问接收数据缓冲区时为什么必须关闭中断？**

   串口设备随时都有可能接收到数据，从而触发中断，执行 SrlIsr 函数读取 RBR 寄存器中的数据并放入接收数据缓冲区中。如果在 SrlRead 函数从接收数据缓冲区中获取数据的同时触发了 RBR 寄存器就绪中断，显然会破坏接收数据缓冲区中的数据。接收数据缓冲区在这里是临界资源，对临界资源的访问需要进行同步。所以，在 SrlRead 函数中访问接收数据缓冲区时必须关闭中断。

   由于 SrlWrite 函数和 SrlIsr 函数对发送数据缓冲区的访问是同步进行的，在 SrlWrite 函数中访问发送数据缓冲区时就不需要关闭中断了。

3. **中断在设备 I/O 中的重要作用和意义**

   使用中断方式可以让进程在等待硬件设备的响应时让出处理器，调度程序会选择其它进程在处理器上继续执行，从而提高处理器的利用率，并支持多道程序和 I/O 设备的并行操作。同时，由于中断方式是异步执行的，所以在访问临界资源时需要进行同步。

4. **Serial.exe 应用程序的参考代码**

   请参见该实验文件夹中的“Serial.c”文件。

5. **实现向串口发送数据功能的参考代码**

   请参见该实验文件夹中的“Serial 参考答案.c”文件。

2.10.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的向串口发送数据功能。可以查看学生编写的源代码，也可以使用学生修改后的 EOS 内核执行 Serial.exe，查看是否能够正常的向串口发送数据。

2.11 磁盘调度算法

2.11.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 EOS 实现磁盘调度算法的机制，如果学生自己阅读和理解相关的源代码比较困难，建议教师可以进行适当的讲解，例如 REQUEST 结构体、IopReceiveRequest 函数、IopProcessNextRequest 函数、IopReadWriteSector 函数和 IopDiskSchedule 函数等。

在开始实验前应该将学生包中的 sstf.c 和 scan.c 源代码文件分发给学生，sstf.c 文件中的 IopDiskSchedule 函数实现了 SSTF 算法，scan.c 文件中的 IopDiskSchedule 函数实现了 SCAN 算法。注意，不要将教师包中的 scan2.c、cscan.c 和 n-step-scan.c 文件分发给学生，该实验要求学生自己编写这些文件中的源代码。

其它需要注意的问题：

1. 不是磁道而是柱面。准确的说，请求中的 Cylinder 域记录的是线程访问的柱面，而不是磁道号。在本实验中使用磁道号这个概念是为了简单，并且保持与操作系统教材一致。以 3.5 英寸 1.44MB 软盘为例，有 2 个盘面，每个盘面有 80 个磁道，每个磁道有 18 个扇区。使用线程要访问的扇区号整除 18，再整除 2，就可以计算出线程访问的柱面，在 IopReceiveRequest 函数中就是这样计算的。
2. 虽然 EOS 使用的软盘只有 80 个柱面，而有些操作系统教材上关于磁盘调度算法的示例所使用的磁道号（柱面）大于 80，但是仍然可以将这些示例数据放入 ConsoleCmdDiskSchedule 函数来进行测试。原因是，EOS 只是在软盘驱动程序中对越界的柱面返回了错误，但是并没有处理这些错误，磁盘调度算法仍然会执行。
3. OS Lab 的“输出”窗口中显示的“平均寻道数”是使用“寻道总数”整除“寻道次数”得到的商(省略余数)，主要是由于 EOS 内核的 C 库函数还不支持浮点数的格式化输出。但是，本实验中的“寻道次数”都是 10 次，所以，可以很容易的计算出浮点的“平均寻道数”。

2.11.2 问题答案及参考代码

1. 只遍历一次请求队列的 SCAN 算法的参考源代码

请参见该实验文件夹中的“scan2.c”文件。

2. CSCAN 算法参考源代码

请参见该实验文件夹中的“cscan.c”文件。

3. N-Step-SCAN 算法参考源代码

请参见该实验文件夹中的“n-step-scan.c”文件。

4. 在执行 SCAN、N-Step-SCAN 磁盘调度算法时，如果在 EOS 控制台中多次输入“ds”命令，调度的顺序会发生变化的原因和解决方法。

SCAN 算法使用了全局变量 ScanInside 记录磁头移动的方向，在第一次执行“ds”命令时，开始磁头是从外向内移动的，结束时磁头变为从内向外移动，所以在下次执行“ds”命令时，开始磁头就继续从内向外移动了。N-Step-SCAN 算法是由于使用了全局变量 SubQueueRemainLength 记录第一个子队列剩余的长度，道理与 SCAN 算法相同。

要解决这个问题，可以在 IopProcessNextRequest 函数中，每次检测到磁盘调度算

法结束工作时，将这些全局变量恢复为默认值。

2.11.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写源代码。可以多准备几组数据用于测试学生编写的磁盘调度算法。对于 N-Step-SCAN 算法，可以使用不同的 N 值来进行测试。

2.12 基本的输入输出

2.12.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。

2.12.2 问题答案及参考代码

目前只实现了将贪吃蛇向右移动、向下移动、向左移动和向上移动的基础功能，请读者在充分理解之前添加的代码的基础上，为贪吃蛇游戏添加下面的功能，使其具有一定的可玩性：

1. **当蛇头“+”移动到屏幕的边缘时，就会在与之相反的边缘出现，继续同方向移动。**
2. **在屏幕上的某些位置出现“#”字符，当蛇头“+”与“#”相遇后，“#”消失，并且在另外一个位置再出现一个“#”。同时，蛇的尾部就多出一个“*”，作为蛇身，吃的“#”越多，蛇身就越长。**
3. **蛇身越长，贪吃蛇移动的速度越快。**
4. **当蛇头“+”撞到蛇身“*”后，结束游戏。**
5. **同时出现2个贪吃蛇，实现双人对战，甚至多人对战。**

略。开放性练习，需读者自己来完成。

2.10.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的在内核中断中控制贪吃蛇移动的源代码。

2.13 扫描 FAT12 文件系统管理的软盘

2.13.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 FAT12 文件系统的技术细节，特别是与 FAT 表和根目录相关的内容。

注意，不要将教师包中的 dir.c 和 sd.c 文件分发给学生，本实验要求学生自己编写这两个文件中的源代码。

2.13.2 问题答案及参考代码

1. **根据 BPB 中的信息计算出其他信息的源代码。**
   请参见教师包中的 sd.c 文件。
2. **输出每个文件所占用的磁盘空间的大小的源代码。**
   请参见教师包中的 dir.c 文件。
3. **在ConsoleCmdScanDisk函数中扫描FAT表时，为什么不使用FAT表项的数量进行计数，而是使用簇的数量进行计数呢？而且为什么簇的数量要从2开始计数呢？**
   对于FAT12文件系统，由于FAT表项有3072项，而实际的簇只有2847个，所以只能使用簇的数量进行计数。由于FAT12文件系统规定FAT表的前两项有固定的用途，他们对应的0、1两个簇号也就不能使用了，所以簇从2开始编号。
4. **在ConsoleCmdScanDisk函数中扫描FAT表时，统计了空闲簇的数量，然后使用簇的总数减去空闲簇的数量做为占用簇的数量，这种做法正确吗？是否还有其他类型的簇没有考虑到呢？**
   这种计算方法不是十分准确，因为还有其他的簇类型，例如坏簇等。

2.13.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的源代码并查看其执行的效果。

2.14 读文件和写文件

2.14.1 准备实验

建议在实验前结合幻灯片向学生讲解预备知识中的内容。建议重点讲解 FAT12 文件系统的结构和特点。

2.14.2 问题答案及参考代码

1. **结合 FAT12 文件系统，说明“文件大小”和“文件占用磁盘空间大小”的区别，并举例说明文件的这两个属性值变化的方式有什么不同。**

   文件占用磁盘空间是以簇为单位的，而文件大小的单位是字节，所以文件大小小于等于文件占用的磁盘空间。当文件增大时，并不一定需要增加磁盘占用空间。例如一个文件当前只有1字节，那么它将占用一个簇的磁盘空间。当文件大小增加到100字节时，它占用的磁盘空间仍然为一个簇。当它的大小增加到超过一个簇的大小时，那么就需要为它增加磁盘空间了。

2. **EOS 应用程序在读写文件时，缓冲区大小设置为 512 的倍数比较合适，说明原因。**

   在 EOS 内核中，读取文件内容都是通过调用 IopReadWriteSector 函数来完成的，而该函数每次最多读取一个扇区（512 字节）的数据，如果 EOS 应用程序设置的缓冲区小于 512 字节，在读取多个扇区时的效率就会较低。例如使用 256 字节的缓冲区读取 2 个扇区的数据需要调用 4 次 IopReadWriteSector 函数，而如果使用 512 字节的缓冲区就只需要调用 2 次。如果要为 EOS 应用程序设置大于 512 字节的缓冲区，也应该使用 512 的倍数。例如使用 768 字节的缓冲区读取 3 个扇区的数据需要调用 4 次 IopReadWriteSector 函数，而如果使用 1024 字节的缓冲区就只需要调用 3 次，所以，如果缓冲区不是 512 的倍数，会增加调用 IopReadWriteSector 函数的次数。

3. **使 FatWriteFile 函数可处理跨越扇区边界的写入数据的参考代码**

   请参见本实验文件夹中的“FatWriteFile 部分实现.c”文件。

4. **使 FatWriteFile 函数可处理跨越多个扇区边界的写入数据的参考代码**

   请参见本实验文件夹中的“FatWriteFile 完整实现.c”文件。同时，该文件中的代码也可以处理一个簇中包含多个扇区的情况。

5. **使用链式分配方式管理磁盘空间的优缺点**

   优点：由于链式分配采用了离散分配方式，从而解决了连续分配带来的碎片问题，可以显著提高磁盘的利用率，并且无需事先知道文件的长度，而是根据文件的当前需要，为它分配必须的盘块。当文件动态增长时，可动态的再为它分配盘块。此外，对文件的增、删、改，也十分方便。

   缺点：链式分配对随机访问是极其低效的，当需要随机访问某个盘块时必须从链表的开始顺序访问。此外，只通过链接指针来将一大批离散的盘块链接起来，其可靠性较差，因为只要其中任何一个指针出现问题，都会导致整个链的断开。

2.14.3 检查实验结果

- 登录到开放实验管理平台查看学生出勤情况及所编写的代码量。
- 检查学生编写的写文件功能。可以查看学生编写的源代码，也可以使用学生修改后的 EOS 内核执行写文件功能，查看是否能够正常的写文件。

附录 A：EOS 核心源代码协议

允许所有人复制和发布本协议文件的完整版本，但不允许对它进行任何修改。

该协议用于控制与之配套的软件，并且管理您使用遵守该协议软件的方法。下面授予您的各项权利受限于该协议。只有当您是合格的教育机构并且从北京英真时代科技有限公司购买了该软件授权时才能够享有这些权利。

在您的教育机构中，您可以为了任何非商业的目的来使用、修改该软件，包括制作合理数量的拷贝。非商业的目的可以是教学、科研以及个人的实验行为。您可以将拷贝发布到机构内部安全的服务器上，并且可以在合格用户的个人主机上安装。

您可以在研究论文、书籍或者其他教育资料中使用该软件的代码片断，或者在以教学和研究为目的的网站、在线公共论坛中发布该软件的代码片断。在您使用的单个代码片断中源代码的数量不能超过 50 行。如您想使用该软件中的大量源代码，请联系 support@tevation.com。

您不能为了商业目的使用或者分发该软件以及从该软件衍生出的任何形式的任何产品。商业目的可以是经营、许可、出售该软件或者为了在商业产品中使用该软件而分发该软件。如果您希望将您的与该软件有关的产品商业化，或者希望与工业伙伴合作研究，您需要联系 sales@tevation.com 来咨询商业授权协议。

您可以为了非商业的目的分发该软件并且修改该软件，但是只能是对于其他该软件的合法用户（例如，将修改的版本分发给其他大学的学生或者教授进行联合学术研究）。只有从北京英真时代科技有限公司购买了该软件授权的合格教育机构，才是合法用户。您不能为该软件或者该软件的衍生产品授予比该协议所提供的更加广泛的权利。

您还必须遵守下面的条款:

1. 您不会从该软件中移除任何版权信息或者布告，也不会对该软件中的二进制部分进行逆向工程或者反编译。

2. 无论您以任何形式分发该软件，您都必须同时分发该协议。

3. 如果您修改了该软件或者创造了该软件的衍生产品，并且分发了修改后的版本或者衍生产品，您需要在被修改文件中的显著位置添加布告来说明您修改的内容和修改日期，这样接收者就会知道他们收到的不是原始的软件。

4. 该软件没有任何担保，包括明示的、暗喻的以及法定的。在适用法律所允许的最大范围内，北京英真时代科技有限公司或其供应商绝不就因使用或不能使用本“软件”所引起的或有关的任何间接的、意外的、直接的、非直接的、特殊的、惩罚性的或其它任何损害赔偿（包括但不限于因人身伤害或财产损坏而造成的损害赔偿，因利润损失、营业中断、商业信息的遗失而造成的损害赔偿，因未能履行包括诚信或相当注意在内的任何责任致使隐私泄露而造成的损害赔偿，因疏忽而造成的损害赔偿，或因任何金钱上的损失或任何其它损失而造成的

损害赔偿）承担赔偿责任，即使北京英真时代科技有限公司或其任何供应商事先被告知该损害发生的可能性。即使补救措施未能达到预定目的，本损害赔偿排除条款将仍然有效。

5. 您不能使用该软件来帮助开发任何为下列目的而设计的软件程序：(a) 对计算机系统有害的或者故意干扰操作的，包括计算机系统中的任何数据和信息；(b) 秘密获取或者维持对计算机系统高级访问权限，有自我繁殖能力，能够在不被发现的情况下执行，包括但不限于所谓的 "rootkit" 软件程序，病毒或者蠕虫。

6. 如果您以任何方式违背了此协议，此协议赋予您的权利就会立即终止。

7. 北京英真时代科技有限公司保留在此协议中明确授予您的权利之外的所有权利。

版本：2008.09.16