
《操作系统》课程实验

2024-2025 学年春季学期

说明

本课程实验共分 6 组，最多任选其中 3 组。每组实验包含若干实验题目，组内最多只能选择 1 题。

不同实验的难度不同，基准分也不同，请同学根据自己的情况加以选择。

计时时，将选择的题目分数相加，满分为 300 分，超过 300 分按照 300 计。

6 组实验题目如下：

实验一：进程间同步/互斥问题

实验二：高级进程间通信

实验三：存储管理

实验四：处理机调度

实验五：文件系统

实验六：驱动程序

提交实验报告的时候需要将实验报告和源程序打包成一个压缩文件，命名为“学号姓名实验 x.rar”或“学号姓名实验 x.zip”，提交至网络学堂对应实验的作业提交窗口。

请勿抄袭作业，**否则 0 分**。

提交截止时间：2025 年 6 月 8 日（校历第 16 周周日）。

实验 1：进程间同步/互斥问题

实验目的：

1. 通过对进程间通信同步/互斥问题的编程实现，加深理解信号量和 P、V 操作的原理；
2. 对 Windows 或 Linux 涉及的几种互斥、同步机制有更进一步的了解；
3. 熟悉 Windows 或 Linux 中定义的与互斥、同步有关的函数。

实验题目：

本实验共有 4 个实验题目，任选其中之一。不同实验的难度不同，基准分也不同，请同学根据自己的情况加以选择。

实验题目	基准分
读者-写者问题	85
哲学家进餐问题	90
睡眠理发师问题	90
银行柜员服务问题	100

操作系统平台可选 Windows 或 Linux，编程语言不限。

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 对提出的问题进行解答；
- 体会或者是遇到的问题。

① 读者-写者问题

问题描述

创建一个包含 n 个线程的进程。用这 n 个线程来表示 n 个读者或写者。允许多个读者同时读一个共享对象，但禁止读者、写者同时访问一个共享对象，也禁止多个写者访问一个共享对象。

附加限制：

- 1、读者优先：无附加限制；
- 2、写者优先：如果一个读者申请进行读操作时已有另一写者在等待访问共享资源，则该读者必须等到没有写者处于等待状态后才能开始读操作。

请分别实现读者优先和写者优先算法。

测试文本格式

每个线程按相应测试数据文件的要求进行读写操作。测试数据文件包括 n 行测试数据，分别描述创建的 n 个线程是读者还是写者，以及读写操作的开始时间和持续时间。每行测试数据包括 4 个字段，各个字段间用空格分隔。

- 第一个字段为一个正整数，表示线程序号
- 第二个字段表示相应线程角色，R 表示读者，W 表示写者
- 第三个字段为一个正数，表示读/写操作的开始时间：线程创建后，延迟相应时间（单位为秒）后发出对共享资源的读/写请求
- 第四个字段为一正数，表示读/写操作的持续时间：线程读写请求成功后，开始对共享资源的读/写操作，该操作持续相应时间后结束，并释放共享资源

下面是一个测试数据文件的例子：

```
1 R 3 5
2 W 4 5
3 R 5 2
4 R 6 5
5 W 5.1 3
```

运行结果显示要求

要求在每个线程创建、发出读写操作申请、开始读写操作和结束读写操作时分别显示一行提示信息，以确定所有处理都遵守相应的读写操作限制。

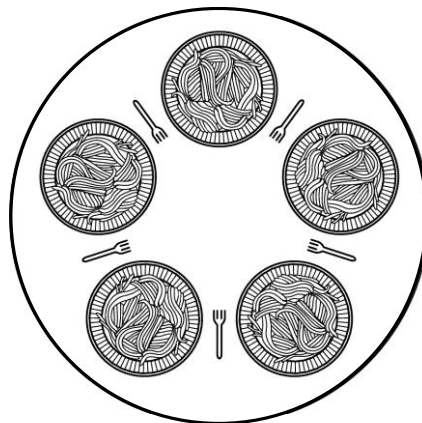
思考题

了解 Windows 或 Linux 中定义的 IPC 函数，比较说明涉及到的几种同步对象。你猜测哪一种操作速度比较快，证实你的想法，并给出一个合理的解释。

② 哲学家进餐问题

问题描述

如下图所示，5 个哲学家围绕一张圆桌而坐，桌子上放着 5 把叉子，每两个哲学家之间放一支；哲学家的动作包括思考和进餐，饥饿时便试图取其左右最靠近他的叉子。



要求：

- (1) 只有拿到两只叉子时，哲学家才能进餐；
- (2) 如果叉子已被别人拿走，则必须等别人进餐完毕才能拿到叉子。

如何保证哲学家们的动作有序进行？

思考题

了解 Windows 或 Linux 中定义的 IPC 函数，比较说明涉及到的几种同步对象。你猜测哪一种操作速度比较快，证实你的想法，并给出一个合理的解释。

③ 睡眠理发师问题

问题描述

理发店里有一位理发师，一把理发椅和 N 把供等候理发的顾客坐的椅子。

如果没有顾客，则理发师便在理发椅上睡觉。当一个顾客到来时，他必须先唤醒理发师。

如果顾客到来时理发师正在理发，则如果有空椅子，可坐下来等；否则离开。

请编程解决这个问题。

思考题

了解 Windows 或 Linux 中定义的 IPC 函数，比较说明涉及到的几种同步对象。你猜测哪一种操作速度比较快，证实你的想法，并给出一个合理的解释。

④ 银行柜员服务问题

问题描述

银行有 n 个柜员负责为顾客服务，顾客进入银行先取一个号码，然后等着叫号。当某个柜员空闲下来，就叫下一个号。

编程实现该问题，用 P、V 操作实现柜员和顾客的同步。

实现要求

1. 某个号码只能由一名顾客取得；
2. 不能有多于一个柜员叫同一个号；
3. 有顾客的时候，柜员才叫号；

-
4. 无柜员空闲的时候，顾客需要等待；
 5. 无顾客的时候，柜员需要等待。

实现提示

1. 互斥对象：顾客拿号，柜员叫号；
2. 同步对象：顾客和柜员；
3. 等待同步对象的队列：等待的顾客，等待的柜员；
4. 所有数据结构在访问时也需要互斥。

测试文本格式

测试文件由若干记录组成，记录的字段用空格分开。记录第一个字段是顾客序号，第二个字段为顾客进入银行的时间，第三个字段是顾客需要服务的时间。

下面是一个测试数据文件的例子：

```
1 1 10
2 5 2
3 6 3
```

输出要求

对于每个顾客需输出进入银行的时间、开始服务的时间、离开银行的时间和服务柜员号。

思考题

1. 柜员人数和顾客人数对结果分别有什么影响？
2. 实现互斥的方法有哪些？各自有什么特点？效率如何？

实验 2：高级进程间通信问题

实验目的：

- 1. 通过对进程间高级通信问题的编程实现，加深理解进程间高级通信的原理；
- 2. 对 Windows 或 Linux 涉及的几种高级进程间通信机制有更进一步的了解；
- 3. 熟悉 Windows 或 Linux 中定义的与高级进程间通信有关的函数。

实验题目：

本实验共有 2 个实验题目，任选其中之一。不同实验的难度不同，基准分也不同，请同学根据自己的情况加以选择。

实验题目	基准分
二元自然数变量函数计算问题	90
快速排序问题	100

操作系统平台可选 Windows 或 Linux，编程语言不限。

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 对提出的问题进行解答；
- 体会或者是遇到的问题。

① 二元自然数变量函数计算问题

问题描述：

设有二元自然数变量函数 $F(m,n) = f(m) + g(n)$ ，其中

$$f(m) = \begin{cases} f(m-1) * m & , m > 1 \\ 1 & , m = 1 \end{cases}$$
$$g(n) = \begin{cases} g(n-1) + g(n-2) & , n > 2 \\ 1 & , n = 1, 2 \end{cases}$$

请编程建立 3 个并发协作进程或线程，它们分别完成计算 $F(m,n)$ 、 $f(m)$ 和 $g(n)$ 。

实验步骤：

- (1) 首先要创建三个线程（或进程），分别执行函数 $F(m,n)$ 、 $f(m)$ 和 $g(n)$ 计算；
- (2) 线程（或进程）之间的通信可以选择下述机制之一进行：
 - 管道（无名管道或命名管道）
 - 消息队列
 - 共享内存
- (3) 通过适当的函数调用创建上述 IPC 对象，通过调用适当的函数调用实现数据的读出与写入；
- (4) 需要考虑线程（或进程）间的同步；
- (5) 线程（或进程）运行结束，通过适当的系统调用结束线程（或进程）。

实验平台和编程语言：

自由选择 Windows 或 Linux。

编程语言不限。

思考题

1. 你采用了你选择的机制而不是另外的两种机制解决该问题，请解释你做出这种选择的理由。
2. 你认为另外的两种机制是否同样可以解决该问题？如果可以请给出你的思路；如果不能，请解释理由。

② 快速排序问题

问题描述：

对于有 1,000,000 个乱序数据的数据文件执行快速排序。

实验步骤：

- (1) 首先产生包含 1,000,000 个随机数（数据类型可选整型或者浮点型）的数据文件；
- (2) 每次数据分割后产生两个新的进程（或线程）处理分割后的数据，每个进程（线程）处理的数据小于 1000 以后不再分割（控制产生的进程在 20 个左右）；
- (3) 线程（或进程）之间的通信可以选择下述机制之一进行：
 - 管道（无名管道或命名管道）
 - 消息队列
 - 共享内存
- (4) 通过适当的函数调用创建上述 IPC 对象，通过调用适当的函数调用实现数据的读出与写入；
- (5) 需要考虑线程（或进程）间的同步；
- (6) 线程（或进程）运行结束，通过适当的系统调用结束线程（或进程）。

实验平台和编程语言：

自由选择 Windows 或 Linux。

编程语言不限。

思考题

1. 你采用了你选择的机制而不是另外的两种机制解决该问题，请解释你做出这种选择的理由。
2. 你认为另外的两种机制是否同样可以解决该问题？如果可以请给出你的思路；如果不能，请解释理由。

实验 3：存储管理问题

实验题目：

本实验共有 3 个实验题目，任选其中之一。不同实验的难度不同，基准分也不同，请同学根据自己的情况加以选择。

实验题目	基准分
文件字节倒放问题	90
AVL 树→红黑树问题	100 + 20 (鼓励分)
动态分区存储管理	100

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 对提出的问题进行解答；
- 体会或者是遇到的问题。

① 文件字节倒放问题

问题描述：

生成一个由随机产生的字符型数据组成大的数据文件（例如，大小 $\geq 1\text{GB}$ ）。将该文件中的所有字节进行倒放，存入原文件，即将文件中的首字节与尾字节对换，次字节与次尾字节对换，以此类推。

编写两个程序，一个程序采用常规的文件访问方法，另一个程序采用内存映射文件方法。请记录两种方法完成字节倒放所需要的时间，并进行比较。

实验环境

操作系统平台可选 Windows 或 Linux，编程语言不限。

函数参考

- Windows: `CreateFileMapping` 创建一个文件映射对象, `MapViewOfFile` 将文件映射对象映射到当前进程的地址空间, `UnmapViewOfFile` 在当前进程的内存地址空间解除对一个文件映射对象的映射;
- Linux: `mmap` 把一个文件映射到当前进程的地址空间, `munmap` 解除内存映射。

思考题

1. 采用常规的文件访问方法时, 改变缓冲区的大小对程序的性能有什么影响? 请用图表描述缓冲区的大小与程序性能之间的关系。
2. 内存映射文件方法和常规的文件访问方法在性能上有什么差异, 试分析其原因。

② AVL 树→红黑树问题

问题描述

在 Windows 的虚拟内存管理中, 将 VAD 组织成 AVL 树。VAD 树是一种平衡二叉树。

红黑树也是一种自平衡二叉查找树, 在 Linux 2.6 及其以后版本的内核中, 采用红黑树来维护内存块。

请尝试参考 Linux 源代码将 WRK 源代码中的 VAD 树由 AVL 树替换成红黑树。

鼓励措施

本实验涉及到操作系统源代码的修改, 难度大, 可能需要较多的时间完成。选做此题的同学可以获得额外的 20 分。

③ 动态分区存储管理

问题描述

分区存储管理有固定分区和动态分区两种方法。固定分区把内存划分为若干个固定大小

的连续分区，每个分区的边界固定；而动态分区并不预先将内存事先划分成分区，当程序需要装入内存时系统从空闲的内存区中，采用不同的分配算法分配大小等于程序所需的内存空间。它有效地克服了固定分区方式中，由于分区内部剩余内存空置造成浪费的问题。

请基于空闲内存分区链表的存储管理，设计一个动态分区存储管理程序，支持包括首次适配法、下次适配法、最佳适配法和最坏适配法在内的不同分区分配算法。

实现要求

1. 维护一个记录已分配内存分区和空闲内存分区的链表；
2. 设计申请、释放函数循环处理用户的请求；
3. 实现首次适配法、下次适配法、最佳适配法和最坏适配法四种分区分配算法；
4. 可视化展示内存使用情况。

实验环境

操作系统平台可选 Windows 或 Linux，编程语言不限。

思考题

基于位图和空闲链表的存储管理各有什么优劣？如果使用基于位图的存储管理，有何额外注意事项？

实验 4：处理机调度

实验题目：

本实验共有 2 个实验题目，任选其中之一。不同实验的难度不同，基准分也不同，请同学们根据自己的情况加以选择。

实验题目	基准分
银行家算法	100
实时系统处理机调度	100

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 对提出的问题进行解答；
- 体会或者是遇到的问题。

① 银行家算法

问题描述

银行家算法是避免死锁的一种重要方法，将操作系统视为银行家，操作系统管理的资源视为银行家管理的资金，进程向操作系统请求分配资源即企业向银行申请贷款的过程。

请根据银行家算法的思想，编写程序模拟实现动态资源分配，并能够有效避免死锁的发生。

实现要求

1. 对实现的算法通过流程图进行说明；
2. 设计不少于三组测试样例，需包括资源分配成功和失败的情况；
3. 能够展示系统资源占用和分配情况的变化及安全性检测的过程；

-
4. 结合操作系统课程中银行家算法理论对实验结果进行分析，验证结果的正确性；
 5. 分析算法的鲁棒性及算法效率。

实验环境

操作系统可选择 Windows 或 Linux，编程语言不限。

思考题

银行家算法在实现过程中需注意资源分配的哪些事项才能避免死锁？

② 实时系统处理机调度

问题描述

实时系统是时间起主导作用的系统。外设给计算机一个刺激，计算机必须在一个确定的时间范围内恰当地做出反应。实时系统中的事件可以按响应方式分为周期性事件和非周期性事件，系统要响应多个周期性事件流和非周期性事件流。

请编程实现实时系统调度中的速率单调调度，最早截止时限优先调度和最小裕度优先三种算法，对多个周期性事件流和非周期性事件流进行响应和处理，并对不同算法的优劣势进行分析。

实现要求

1. 为每一个事件流创建响应的进程；
2. 输入包含多个周期性事件流和非周期性事件流；
3. 至少设计一组输入样例，使得三种调度算法都满足绝对的截止时间；
4. 至少设计一组输入样例，使得至少一种调度算法不满足绝对的截止时间。

实验环境

操作系统平台可选 Windows 或 Linux，编程语言不限。

测试文本格式

测试文件由若干记录组成，记录的字段用空格分开。第一行记录事件流进入的最大时间，所有周期性事件流的进入时间都小于此时间。从第二行开始，记录第一个字段是事件流编号，第二个字段为是否为周期性事件流（1 为是，0 为否）。对于周期性事件流，第三、第四、第五字段分别为其第一次进入时间，周期和每次运行所需时间；对于非周期性事件流，第三、第四、第五字段分别为其进入时间，截止时间和运行所需时间。

下面是一个测试数据文件的例子：

200

1 1 0 50 20

2 1 0 25 5

3 0 5 35 10

4 0 15 60 5

输出要求

对于每个事件流，依次输出其进入时间，截止时间，响应开始时间和结束时间。

思考题

三种调度算法有何优劣？各自适用于什么情境？

实验 5：文件系统问题

实验题目：

本实验共有 3 个实验题目，任选其中之一。不同实验的难度不同，基准分也不同，请同学根据自己的情况加以选择。

实验题目	基准分
磁盘 IO 问题	80
FAT 文件系统实现	95
RAID 实现	100

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 体会或者是遇到的问题。

① 磁盘 IO 问题

问题描述：

通过针对磁盘进行 I/O 实验，了解与掌握直接访问磁盘扇区的方法。

要求实现三个函数：

- 函数 `physicalDisk` 判定逻辑驱动器 X 中磁盘的基本信息；
- 函数 `sectorRead` 根据给定的物理扇区号读取磁盘的扇区；
- 函数 `sectorDump` 查看磁盘的内容并把磁盘上得到的信息输出到标准输出流中。

编写一个程序调用 `physicalDisk`、`sectorRead` 和 `sectorDump` 三个函数，验证其正确性。

实验环境：

操作系统平台可选 Windows 或 Linux，编程语言不限。

Windows 实现函数参考：

1. 打开驱动器只需要调用 `CreateFile` 函数即可，例如打开 E 盘：

```
hDevice = CreateFile("\\\\.\\E:",           // drive to open
                    0,                   // no access to the drive
                    FILE_SHARE_READ | FILE_SHARE_WRITE, // share mode
                    NULL,                // default security attributes
                    OPEN_EXISTING,        // disposition
                    0,                    // file attributes
                    NULL);                // do not copy file attributes
```

2. 读取驱动器信息可以通过 `DeviceIoControl` 函数。调用 `DeviceIoControl` 函数时传递 `IOCTL_DISK_GET_DRIVE_GEOMETRY` 控制代码，即可返回指向 `DISK_GEOMETRY` 结构的指针：

```
BOOL DeviceIoControl(
    (HANDLE) hDevice,           // handle to device
    IOCTL_DISK_GET_DRIVE_GEOMETRY, // dwIoControlCode
    NULL,                       // lpInBuffer
    0,                           // nInBufferSize
    (LPVOID) lpOutBuffer,       // output buffer
    (DWORD) nOutBufferSize,     // size of output buffer
    (LPDWORD) lpBytesReturned,  // number of bytes returned
    (LPOVERLAPPED) lpOverlapped // OVERLAPPED structure
);
```

`DISK_GEOMETRY` 是一个重要的结构，反映了磁盘的几何布局：

```
typedef struct _DISK_GEOMETRY {
    LARGE_INTEGER Cylinders;
```

```
MEDIA_TYPE MediaType;

DWORD TracksPerCylinder;

DWORD SectorsPerTrack;

DWORD BytesPerSector;

} DISK_GEOMETRY;
```

3. 读取扇区时，要注意每个扇区的长度，并根据每个扇区的长度*逻辑扇区号来移动文件指针。将扇区内容输出到屏幕时，应该将每字节按照两位 16 进制数输出，排列整齐。

```
SetFilePointer(hDevice,

    Disk_info.BytesPerSector*logicSectorNumber,

    NULL,

    FILE_BEGIN);

ReadFile(hDevice,

    buffer,

    disk_info.BytesPerSector,

    &numberofread,

    NULL);
```

Linux 实现函数参考：

1. 打开驱动器只需要调用 open 函数即可，函数原型：

```
int open (const char *pathname, int flags , ...);
```

pathname 为要打开的文件名，因为 Linux 将设备描述为文件，所以可以在目录/dev 中找到所有设备文件，例如 IDE（Integrated Drive Electronics）硬盘驱动器/dev/hda 和 /dev/hdb。flags 必须包含 O_RDONLY、O_WRONLY 或 O_RDWR 的其中之一，其他标志可选。

例如，打开 IDE 硬盘驱动器/dev/had：

```
fd = open("/dev/hda ", O_RDONLY);
```

2. 读取驱动器信息可以通过 ioctl 函数，函数原型：

```
int ioctl(int fd, int request, /* arg */ ...);
```

第一个参数 fd 是函数 open() 返回的文件描述符，用于指称具体设备。与 Windows 对应的系统调用 DeviceIOControl 不同，ioctl 的输入参数列表并不固定。它取决于 ioctl 进行

何种请求，以及请求参数有何说明。

获取扇区大小使用 `BLKSSZGET` ioctl，获取磁盘布局使用 `HDIO_GETGEO_BIG` 或 `HDIO_GETGEO` ioctl，获取磁盘大小使用 `BLKGETSIZE64` 或 `BLKGETSIZE` ioctl。

3. 读取扇区时，要注意每个扇区的长度，并根据每个扇区的长度*逻辑扇区号来移动文件指针。将扇区内容输出到屏幕时，应该将每字节按照两位 16 进制数输出，排列整齐。

移动文件指针，可以使用 `lseek` 函数，读设备可以使用 `read` 函数。

② FAT 文件系统实现

问题描述

通过针对 FAT 文件系统进行 I/O 实验，了解与掌握 FAT 文件系统。

要求实现一个简单的 FAT 文件系统管理器，具有如下功能：

- 列目录
- 改变文件名
- 删除文件
- 复制文件

实现提示

- 文件列表：顺序检查文件名，第一个字母非空格也非 `0xe5`(表示删除)，即输出其主文件名，扩展名，文件长度等信息
- 文件重命名：首先检查新文件名是否已经存在，否则搜索旧文件名，找到之后修改之
- 文件复制：首先检查新文件名是否已经存在，否则将旧文件的文件列表项复制到空白项，更改文件名和首文件簇。然后将旧文件的文件簇序列的内容依次复制到新文件的文件簇，同时更新 FAT
- 文件删除：先将根目录文件表项清空，然后将 FAT 中的文件簇链清空

实验环境

操作系统平台为 Windows 或 Linux，编程语言不限。

需要注意的问题

可以调用 Windows 的 `CreateFile` 函数或者 Linux 的 `open` 函数打开一个磁盘驱动器。对于打开的磁盘驱动器，需要先读取第 1 扇区，判断是否是 FAT 文件系统，如果不是，请给出出错提示，并退出程序；如果是 FAT 文件系统，则从 BPB 中提取磁盘参数。

只允许进行直接磁盘 IO，不允许使用 Windows 系统下的 `GetCurrentDirectory`、`CreateDirectory`、`FindFirstFile`、`DeleteFile` 等 API 函数，或者 Linux 系统下的 `mkdir`、`rmdir`、`link`、`unlink` 等系统调用。

为避免损失，请不要在硬盘上进行实验。

③ RAID 实现

问题描述

RAID（Redundant Array of Independent Disks）是利用一台磁盘阵列控制器，统一管理和控制一组磁盘驱动器，组成一个速度快、可靠性高、性能价格比好的大容量磁盘系统。

要求实现一个简单的 RAID 工具，具有如下功能：

- 创建 RAID0、RAID1、RAID3、RAID5 阵列，并可以读、写文件
- 针对上述磁盘阵列，模拟其中 1 块磁盘损坏时读取数据的过程，并利用 1 块新磁盘重建阵列和恢复数据

实现提示

可以利用虚拟机软件，新建一个虚拟机，并在虚拟机软件中添加若干块虚拟磁盘，以模拟多块磁盘的环境。从虚拟机软件中删除一块磁盘，模拟磁盘出问题的状态。

实验环境

操作系统平台为 Windows 或 Linux，编程语言不限。

需要注意的问题

不可以调用已有的软 RAID 工具或使用 RAID 阵列卡实现，也不能抄袭已有的软 RAID 工具的代码。

为避免损失，请不要在硬盘上进行实验。

实验 6：驱动程序问题

实验题目：

本实验共有 2 个实验题目，任选其中之一。

实验题目	基准分
Ramdisk 驱动程序开发	90
管道驱动程序开发	100

实验报告内容要求：

- 写出设计思路和程序结构，并对主要代码进行分析；
- 实际程序运行情况；
- 体会或者是遇到的问题。

① Ramdisk 驱动程序开发问题

问题描述

Ramdisk（内存盘）实际上是在系统的内存中划出一块空间当作磁盘使用。内存的存取速度远远大于机械磁盘，所以 Ramdisk 肯定要比机械的磁盘快得多。随着计算机硬件技术的快速发展，目前人们使用的计算机硬件配置越来越强大，特别是内存越来越大，这样就为用内存代替磁盘提供了可能。

在本实验中，通过编写设备驱动程序实现 Ramdisk，可以在其上创建文件系统，实现常规的文件操作。

在微软的网站上，提供了一个 Ramdisk 驱动程序的实例，包括源代码和说明文档，网址为 <http://support.microsoft.com/kb/257405/zh-cn>，请登录该网址，下载相应的文件。

要求

1. 在自己的机器上使 Ramdisk 驱动程序的实例正确运行。

2. 对 Ramdisk.h 文件中的主要数据结构进行分析。
3. 对 Ramdisk.c 文件中的主要函数功能进行分析。
4. 对 Pnp.c 文件中的主要函数功能进行分析。

实验环境：

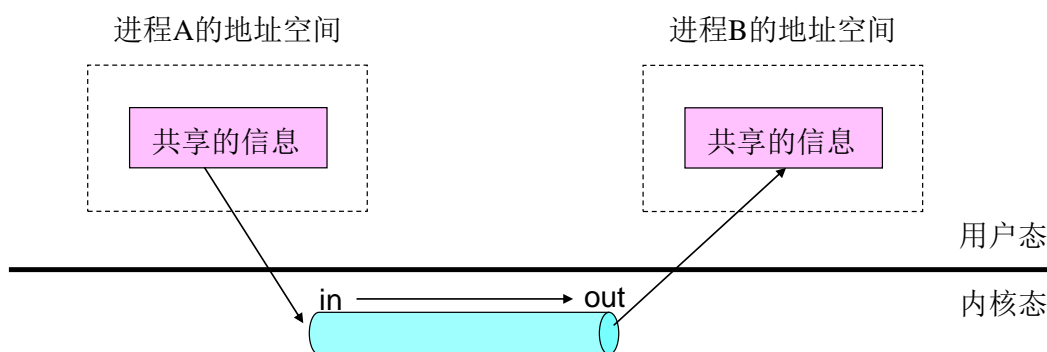
操作系统平台 Windows，编程语言不限。

②管道驱动程序开发

问题描述：

管道是现代操作系统中重要的进程间通信（IPC）机制之一，Linux 和 Windows 操作系统都支持管道。

管道在本质上就是在进程之间以字节流方式传送信息的通信通道，每个管道具有两个端，一端用于输入，一端用于输出，如下图所示。在相互通信的两个进程中，一个进程将信息送入管道的输入端，另一个进程就可以从管道的输出端读取该信息。显然，管道独立于用户进程，所以只能在内核态下实现。



在本实验中，请通过编写设备驱动程序 `mypipe` 实现自己的管道，并通过该管道实现进程间通信。

你需要编写一个设备驱动程序 `mypipe` 实现管道，该驱动程序创建两个设备实例，一个针对管道的输入端，另一个针对管道的输出端。另外，你还需要编写两个测试程序，一个程序向管道的输入端写入数据，另一个程序从管道的输出端读出数据，从而实现两个进程间通过你自己实现的管道进行数据通信。

实验环境：

操作系统平台可选 Windows 或 Linux，编程语言不限。