1.1&1.2 第一周

1.1

1. What are the two main functions of an operating system?

课本摘抄：

1.providing application programmers (and application programs, naturally) a clean

abstract set of resources instead of the messy hardware ones

2.managing these hardware resources

我的理解：

1.给用户提供简洁的接口，避免直接使用较为复杂的硬件

2.管理好各种计算机资源(内存，CPU，磁盘等等)

老师之后评讲的答案:

1）为应用程序提供一个资源抽象及。为应用程序提供一个资源的抽象，即操作系统给用户提供了一个可扩展的机器，通过对底层的抽象，对外提供各种接口支持扩展；

2）管理各种软硬件资源。对资源使用计数、控制资源分配等。

2. What is the difference between timesharing and multiprogramming systems?

课本摘抄：

Timesharing is a variant of multiprogramming

Multiprogramming: Maximize processor use

Timesharing: for quick response time for users

我的理解：

多道是指一个用户可以在内存里存放多个作业，一个个批处理，当某个作业在使用CPU时，另一个作业可以使用IO, 而分时最早是在Multics系统上运用，它想让多个用户共同参与计算，由于完全处理完一个任务等待时间过长，每个用户可以使用这个系统一段时间，也就是分时。分时也是一种多道。

老师之后评讲的答案:

1） 在分时系统中，多个用户共享计算机资源（分时使用CPU时间），他们可以使用自己的终端同时访问计算机，并执行他们的任务。

2） 多道程序设计是指操作系统允许同时运行多个用户程序。 所有分时系统都是多道程序设计系统，但并非所有多道程序设计系统都是分时系统。

10. What is the difference between kernel and user mode? Explain how having two distinctmodes aids in designing an operating system.

课本摘抄：

Kernel mode: has complete access to all the hardware and can execute any instruction the machine is capable of executing

User mode: where the rest of the software runs, only a subset of machine instructions are available

老师之后评讲的答案:

CPU提供两种执行模式：内核态和用户态

1）CPU在内核态模式下（内核态），可以执行其指令集中任何指令，执行硬件的各种功能；CPU在用户模式下（用户态），只能执行部分指令，使用部分功能。

2）CPU采用两种模式的优点：用户的程序任务在用户态运行，OS拒绝用户程序访问关键指令，这样使得操组系统内核受到保护，内核指令不得非法运行。

11. Which of the following instructions should be allowed only in kernel mode?(a) Disable all interrupts.(b) Read the time-of-day clock.(c) Set the time-of-day clock.(d) Change the memory map.

(a) (c) (d)

17. **What is a trap instruction? Explain its use in operating systems.**

原文摘抄：

The TRAP instruction switches from user mode to kernel mode and starts the operating system.

我的理解：

Trap指令可以让用户调用操作系统的服务，从用户模式转成内核模式，是一种特殊的命令。

老师之后评讲的答案:

1） 陷阱指令是处理陷阱的指令，是由于系统调用引起处理机中断的指令，用于操作系统查看CPU的内部事件。

2）在操作系统中，陷阱指令将一个处理器的执行模式从用户态切换到内核态，允许用户程序调用操作系统内核中的函数。

25. **What is the essential difference between a block special file and a character special file?**

原文摘抄：

Special files: They are provided in order to make I/O devices look like files. That way, they can be read and written using the same system calls as are used for reading and writing files. Two kinds of special files exist: block special files and character special files.

Block special files: They are used to model devices that consist of a collection of randomly addressable blocks, such as disks.

character special files: They are used to model printers, modems, and other devices that accept or output a character stream.

老师之后评讲的答案:

1） 块专用文件包含被编号的块，每一块都可以独立地读取或者写入，并且可以定位于任意块，进行读出或写入。块文件通常类似于磁盘设备（在数据可以被访问的地方赋予一个块号，意味着同时设定了一个块缓存）。

2） 字符专用文件不是以块为单位，而是以字符为单位读写，字符文件I/O没有缓存，而块文件则有缓存，为了缓解速度。

1.2

1. What is the purpose of system calls?

原文摘抄：

To obtain services from the operating system, a user program must make a system call, which traps into the kernel and invokes the operating system.

我的理解：

有些指令需要使用内核模式才能执行，用户程序为了使用操作系统，就得有system call。

老师之后评讲的答案:

用户程序不能直接访问内核模式提供服务，为了让用户级进程可以请求操作系统所提供的服务，特权指令只能在核心态处理，因此，用户程序通过使用系统调用就可以使用内核提供的服务。

2.What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

原文摘抄：

Advantage: the designers have a choice where to draw the kernel-user boundary.

Disadvantage: Traditionally, all the layers went in the kernel, but that is not necessary. In fact, a strong case can be made for putting as little as possible in kernel mode because bugs in the kernel can bring down the system instantly.

我的理解：

这一段在微内核处，分层的方式可以让设计者划分好用户和内核的边界，但是某些内核中的错误可能会拖累系统。

老师之后评讲的答案:

1） 优点：使得构造和调试的简单化，每一层功能是限定的，调试时不用考虑其他层次的问题。每层为较高层隐藏了一定的数据结构、操作和硬件的存在。较高层利用较低层所提供的功能来实现，无需关心其具体实现方法。

2） 缺点：效率较低，每层的信息传递需要额外的开销。例如，当一个用户执行I/O操作时，执行系统调用，并陷入I/O层，I/O层会调用内存管理层，内存管理层又调用CPU层，最后传递给硬件。每层都会增加额外开销（如参数传递等）。

3.What is the main advantage of the microkernel approach to system design? What are the disadvantages of using the microkernel approach?

原文摘抄：

Advantage: to achieve high reliability by splitting the operating system up into small, well-defined modules, only one of which—the microkernel—runs in kernel mode and the rest run as relatively power less ordinary user processes.

Disadvantage: Bug density depends on module size, module age, and more, but a ballpark figure for serious industrial systems is between two and ten bugs per thousand lines of code. This means that a monolithic operating system of five million lines of code is likely to contain between 10,000 and 50,000 kernel bugs. The system has many restrictions limiting the power of each process.

我的理解：

微内核将操作系统分为小的模块来提高可靠性，一个模块的崩溃不会造成整个系统的死机。但是代码错误的密度却取决于模块的大小。

老师之后评讲的答案:

优点通常包括以下几点

a）增加一个新的服务不需要修改内核

b) 在用户模式中比在内核模式中更安全、更易操作

c) 每个内核设计简单，也使得操作系统更加可靠，操作系统服务通过使用进程间通信机制在微内核中相互作用，例如发送消息。这些消息由操作系统传输。

微内核的缺点：

1） 通过进程通信的方式交换数据或者调用系统服务,而不是直接使用系统调用,造成操作系统额外的开销。

2） 频繁使用的系统服务，例如，网络收发数据, 进程上下文切换会造成一定开销。

2.1 第二周

1. In Fig. 2-2, three process states are shown. In theory, with three states, there could be six transitions, two out of each state. However, only four transitions are shown. Are there any circumstances in which either or both of the missing transitions might occur?

还有两种状态是从阻塞态到运行态，从就绪态到阻塞态。从阻塞态到运行态可能是可行的，当某个进程从阻塞态复原时，若是此时CPU并没有处理其他的进程，或许可以跳过就绪态处理这个进程；而从就绪态到阻塞态我觉得不太可能，因为就绪态就是在一个队列中等待处理，没有处理的时候不会出现能使进程阻塞的原因。

老师之后评讲的答案:

1） 当系统只有一个进程时，从阻塞到运行状态的转换是可能会发生的，例如。某个进程因为发生了I/O请求产生阻塞，而且当前I/O请求已经结束，如果此时CPU空闲，该进程就可以从阻塞态直接转到运行态。

2） 另外一种转换，是从就绪态到阻塞状态，这是不可能的，也是不必要的。一个就绪进程如果不做任何事情，是不可能，也没必要直接转到阻塞状态，必须要经过运行态后才能进入阻塞。

2. Suppose that you were to design an advanced computer architecture that did process switching in hardware, instead of having interrupts. What information would the CPU need? Describe how the hardware process switching might work.

至少需要一个元件来提醒进程的切换，比如寄存器之类，当需要上下文切换时，寄存器的状态改变，或许可以指向下一个进程的地址，然后保存这个进程的信息，调入下一个进程的信息。

老师之后评讲的答案:

CPU应该有一个寄存器存放当前进程表项的指针。当I/O结束之后，CPU将把当前的机器状态存入到当前进程表项中。然后，将转到中断设备的中断向量，读取另一个过程表项的指针（服务例程），然后，就可以启动这个进程了。

3. On all current computers, at least part of the interrupt handlers are written in assembly language. Why?

原文摘抄：

Actions such as saving the registers and setting the stack pointer can not even be expressed in high-level languages such as C, so they are performed by a small assembly-language routine, usually the same one for all interrupts since the work of saving the registers is identical, no matter what the cause of the interrupt is.

我的理解：

高级语言不能被寄存器这样的硬件设备解释，等到高级语言转成汇编再来处理中断，就会造成时间的浪费，所以中断处理程序有一部分是汇编语言写的。

老师之后评讲的答案:

通常，高级语言不允许访问CPU硬件，而对于中断处理程序而言，访问硬件是必需的，例如，中断处理程序可能需要禁用和启用某个特定设备的中断服务，或者处理进程堆栈区的数据。另外，中断服务程序需要尽快地执行。

4. When an interrupt or a system call transfers control to the operating system, a kernel stack area separate from the stack of the interrupted process is generally used. Why?

一些用户程序不能有足够的堆栈空间，若是内核没有单独的堆栈，可能会导致操作系统的运行崩溃。另外，将内核数据留在用户空间会造成数据的不安全。

老师之后评讲的答案:

内核使用单独的堆栈，主要原因是：

1）内核堆栈与用户程序堆栈分离，可以使得用户程序具有足够的堆栈空间，有利于用户任务的顺利完成。

2）如果内核将数据保留在用户空间，系统调用之后，留在用户空间中的信息，可能就会泄露，出现安全问题。

7. If a multithreaded process forks, a problem occurs if the child gets copies of all the parent's threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem ever occur in single-threaded processes?

这种情况应该不会发生。单进程的父进程在键盘上阻塞时，子进程就不能被创造。

老师之后评讲的答案:

不会。如果单线程进程中，原始线程正在等待键盘输入，就不能创建子进程，必须等待上一个线程完成后才能创建子进程。

3.1 第三周

4. Consider a swapping system in which memory consists of the following hole sizes in

memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB.

Which hole is taken for successive segment requests of

(a) 12 MB

(b) 10 MB

(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

first fit: a->20MB, b-> 10MB, c-> 18MB

best fit: a-> 12MB, b-> 10MB, c->9MB

worst fit: a->20MB, b->18MB, c->15MB

next fit: a->20MB, b-> 18MB, c-> 9MB

5. What is the difference between a physical address and a virtual address?

虚拟地址(也称逻辑地址)是CPU给程序的地址，是从0开始的，而物理地址是程序所占内存的实际地址，通过基址寄存器将逻辑地址映射到物理地址。

老师之后评讲的答案:

（1）物理地址。实际内存空间使用物理地址，内存芯片上地址

（2）虚拟地址是指进程的地址空间的逻辑地址。因此，具有32位字的计算机可以生成高达4 GB的虚拟地址，而不管该计算机的内存是否大于或小于4 GB。

6. For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4-KB page and for an 8 KB page: 20000, 32768, 60000.

4kb: 4kb = 2^{12} bytes = 4096 bytes

20000 / 4096 = 4, 2000 % 4096 = 3616 页号为4 偏移量为3616

32768 / 4096 = 8, 32768 % 4096 = 0 页号为8 偏移量为0

60000 / 4096 = 14, 2000 % 6096 = 2656 页号为14 偏移量为2656

8kb: 4kb = 2^{13} bytes = 8192 bytes

20000 / 8192 = 2, 2000 % 8192 = 3616 页号为2 偏移量为3616

32768 / 8192 = 4, 32768 % 8192 = 0 页号为4 偏移量为0

60000 / 8192 = 7, 2000 % 8192 = 2656 页号为7 偏移量为2656

7. Using the page table of Fig. 3-9, give the physical address corresponding to each of the following virtual addresses:

(a) 20

(b) 4100

(c) 8300

(a) 8192 + 20 = 8212

(b) 4100（由表直接映射）

(c) (8300-8192) + 1024 * 24 = 24684

13. If an instruction takes 1 nsec and a page fault takes an additional n nsec, give a formula for the effective instruction time if page faults occur every k instructions.

$(k+N)/k = 1 + N/k\ ns$

24. A machine has 48-bit virtual addresses and 32-bit physical addresses. Pages are 8 KB. How many entries are needed for a single-level linear page table?

物理页面数量：$2^{32}/2^{13} = 2^{19}$, 虚拟页面数量$2^{48}/2^{13} = 2^{35}$

老师之后评讲的答案:

物理内存是4GB，页面数量是4GB/8KB=2^19项，页面偏移量需要2^13位，页表域总共35位。

3.2 第四&五周

1.Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

当在地址映射的时候，如果发现所在页空缺不在内存中，就会发生缺页中断。

首先操作系统会检查该页是否为合法的内存访问空间，如果不是，则会终止。如果合法的话。则会运用某种算法找到一个空闲页框，将该页与空闲页框交换。当交换完成，重置页表，并重新启动中断的指令。

老师之后评讲的答案:

地址映射过程中，在页表中发现所要访问的页不在内存，则产生缺页中断。缺页中断是在指令执行期间，发现所要访问的指令或数据不在内存时所产生和处理的

操作系统接到此中断信号后，就调出缺页中断处理程序，根据页表中给出的外存地址，将该页调入内存，使作业继续运行下

缺页中断的过程：

1. 如果有对页面引用，第一次引用交给操作系统，产生缺页中断。
2. 检查该进程的内页表，看看该引用是否是合法的内存访问空间,如果不合法，终止该进程；

如果合法，但是没有调入到内存，需要将该进程放入该页中。

3. 查找一个空闲页框.
4. 将该页与空闲页框交换。调度磁盘操作来读取请求的页，将其放入新的分配的页框中。
5. 当磁盘读取操作完成时，重新设置该进程的页表，表明该页在内存中。validation bit = 1（重新设置页表，把位设为1）.
6. 重新启动中断的指令。该进程现在能够访问该页，因为它一直在内存中。

2 Consider the page table shown in the following Figure for a system with 12-bit virtual and physical addresses and with 256-byte pages. The list of free page frames is D, E, F (that is, D is at the head of the list, E is second, and F is last).

Convert the following virtual addresses to their equivalent physicaladdresses in hexadecimal. All numbers are given in hexadecimal. (Adash for a page frame indicates that the page is not in memory.)

• 9EF

• 111

• 700

• 0F F

9EF = 1001 1110 1111　1001 = 9,对应0号页框， 物理地址为0000 1110 0111 为0EF

111 = 0001 0001 0001　0001 = 1,对应2号页框， 物理地址为0010 0001 0001 为211

700 = 0111 0000 0000　0111 = 7,对应free页D， 物理地址为1110 0000 0000 为D00

9FF = 0000 1111 1111　0000 = 0,对应free页E， 物理地址为1110 1111 1111 为EFF

***3)*Consider the following page reference string:**

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement

algorithms, assuming three and five frames?

Remember that all frames are initially empty, so your first unique pages

will cost one fault each.

• LRU replacement

• FIFO replacement

• Optimal replacement

	LRU	FIFO	Opimal
3页	15	16	11
5页	8	10	7

4． A certain computer provides its users with a virtual memory space of232 bytes. The computer has 222 bytes of physical memory. The virtualmemory is implemented by paging, and the page size is 4,096 bytes.

A user process generates the virtual address 11123456. Explain howthe system establishes the corresponding physical location. Distinguishbetween software and hardware operations

一页大小为2^12, 因此页表大小为2^20， 前20bits表示页表中的偏移量，后12bits表示在页内的偏移量

对于11123456，前面的0x11123表示页表偏移量，后面0x456表示页内偏移量

（5）Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.

Assuming demand paging with three frames, how many page faults

would occur for the following replacement algorithms?

• LRU replacement

• FIFO replacement

• Optimal replacement

LRU经过计算，只有第5次，第11次没有发生缺页，共缺页18次

FIFO经过计算，只有第5次，第7次，第11次没有发生缺页，共缺页17次

Optimal经过计算，第3,5,11,12,14,19,20页，共缺页13次

2.2 第六周

(1) What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

用户线程	内核线程
内核不知道用户线程存在	内核知道内核线程的存在
在没有内核支持的情况下管理用户线程	内核线程由OS管理
用户级线程由线程库调度	内核线程由内核调度
用户线程创建速度快	内核进程速度慢

内核级线程优于用户级线程的情况：

①内核是单线程的(执行阻塞系统调用的任何用户级线程都将导致整个进程阻塞，即使其他线程在应用程序中可以运行)

②多处理器环境(内核级线程可以同时在不同的处理器上运行，而进程的用户级线程只在一个处理器上运行，即使多个处理器可用)

用户级线程优于内核级线程的情况：

内核是时间共享的(在时间共享系统中，上下文切换频繁发生。内核级线程之间的上下文切换开销很高，几乎与进程相同，而用户级线程之间的上下文切换与内核级线程相比几乎没有开销)

老师之后评讲的答案:

a. 用户级线程是无需内核感知的线程, 内核级线程需要操作系统内核感知.

b. 系统可以采用映射 M:1 or M:N,用户级线程与内核级线程就联系起来了.用户级线程的上下文切换与内核级线程的上下文切换类似，但是用户级线程的切换需要依赖于线程库与用户级线程与内核级线程之间的映射关系。

c. 内核级线程无需与进程相关联,而每个用户级线程需要与一个进程相关联

d. 内核级线程比用户级线程更难维护，因为它们必须用一个内核固定的数据结构表示

(2) What resources are used when a thread is created? How do they differ from those used when a process is created?

创建一个进程需要分配进程控制块(PCB),是较大的数据结构，创建一个线程需要分配一个小的数据结构来控制寄存器的设置，以及堆栈和优先级

老师之后评讲的答案:

创建一个线程需要分配一小段数据结构(TCB)来维护寄存器组，栈与优先级、调度等信息。

进程创建的时候需要分配PCB，这是一个相当大的数据结构。PCB中包含了内存的映射，打开文件的列表与环境变量。分配和管理内存映射通常是最费时的。

因此，由于线程比进程小很多，所以创建线程需要的资源会比创建进程的小很多。

(3) Which of the following components of program state are shared across threads in a multithreaded process?

a. Register values

b. Heap memory

c. Global variables

d. Stack memory

b c

(4) Compare the cost of thread creation and process creation (with evidence!)

创建进程的时间是大于线程的。

evidence:每个进程都有自己的地址空间,而同一进程内的线程共享进程的地址空间，创建一个新的进程要耗费时间来为其分配系统资源(PCB)，而创建一个新的线程花费的时间要少得多

老师之后评讲的答案:

创建线程的所需要的时间比创建进程少很多

进程：当进程开始时，主要开销在: allocate address space（分配地址单元空间）、establish numerous data tables（创建多个数据表）

线程：由于进程中的线程可以互相通用地址空间，因此他们开销只有: share most of the data （共享数据）。So it takes less space to start a thread than process（比进程需要的空间少）

(5) Describe the advantages and disadvantages of various threading model.

①单线程服务器编程模型： 是最简单的服务器模型；服务器系统资源消耗较小，但并发能力低，容错能力差

②多线程服务器编程模型：支持对多个客户端并发响应，处理能力得到大幅提高，有较大的并发量； 服务器系统资源消耗量较大，而且多线程之间会产生线程切换成本

老师之后评讲的答案:

Many-to-One model：

Advantages：线程管理在用户空间完成，所以它的效率比较高

Disadvantages：但是如果一个线程调用了导致阻塞的系统调用的话，那么将阻塞整个进程。而且，因为一次只有一个线程可以访问内核，所以在多处理机环境中多个线程不能够并发执行。

2.3&2.4 第七&八周

2.3

1.进程间通信有哪几种方式？简述每一种方式。

①共享存储器系统：映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。

②消息传递方式： 进程间的数据交换是以格式化的消息(message)为单位，消息传递方式具有进程之间通信和协同的机制，消息系统-进程之间相互通信，而不使用共享变量

③Pipe管道通信：是指用于连接读进程、写进程之间通信的共享文件，又名pipe文件，可以以字符流形式向管道(共享文件)输入数据，然后从管道中接收(读)数据。

2.举例说明什么是同步通信？什么是异步通信?

同步通信是指两者具有等待关系，比如对一个缓冲区的互斥访问，异步通信是指两者不需要集合点，比如给某一进城发送消息，如果不会被阻塞，还可以继续发送消息。

老师之后评讲的答案:

同步和异步通信: 同步通信允许发送者和接收者之间有一个集合点。异步通信不需要集合点，消息可以异步传递。异步通信一般不会产生阻塞。因此，消息传递系统，往往提供两种形式的同步,常采用结合策略。

如果生产进程将产品放置于缓存中后，等待消费进程从缓存中取出该产品后再进入产品的生产循环。这样生产进程和消费进程就是同步的。

如果生产进程生产出产品后放置于缓存，消费进程是不是从缓存中取走产品并不关心。依旧开始循环生产进程。这样生产进程和消费进程就是异步的。

如果两个进程要交换结构较为复杂的信息，操作系统的消息缓冲区也是系统共享资源，多个进程使用它时，必须同步。

3.什么是管道？有几种类型？分别有什么特点?

管道运输是指用于连接读进程、写进程之间通信的共享文件，又名pipe文件，可以以字符流形式向管道(共享文件)输入数据，然后从管道中接收(读)数据。

分为普通管道和命名管道两种方式。

①普通管道: 父子进程之间使用的通信方式,允许以标准的生产者-消费者风格进行通信,并且是是单向的

②命名管道: 非父子关系的进程之间的通信方式,比普通管道功能更强大,不仅通信是双向的,而且有几个进程可以使用命名管道进行通信.

2.4

1. **Illustrate how a binary semaphore can be used to implement mutual exclusion among n processes.（使用二值信号量实现互斥）举例说明**

```
//mutex初值为1
do {
wait(mutex);
critical section
signal(mutex);
remainder section
} while (true);
```

2.**下面是两个并发执行的进程,它们能正确执行吗?若不能.请举例说明,并修改之.**

```
int x;
Process P1:
void fp1( )
{
    int y,z;
    x=1;
    y= 0;
    if (x>=1)
    y=y+1;
    z=y;
}

Process P2:
void fp2( )
{
    int t,u;
    x=0;
    t= 0;
    if(x<=1) t=t+2;
    u=t;
 }
```

可能不会，当进程1把x改成1之后，在使用x之前，可能进程2会将x改成0，这样会导致if判断出错。

可以引入信号量机制实现互斥：

```
int x;
int mutex = 1;
Process P1:
void fp1( )
{
    int y,z;
    P(muutex);
    x=1;
    y= 0;
    if (x>=1)
    y=y+1;
    z=y;
    V(muutex);
}
```

```
Process P2:
void fp2( )
{
    int t,u;
    P(muutex);
    x=0;
    t= 0;
    if(x<=1) t=t+2;
    V(muutex);
    u=t;
 }
```

3.**“生产者-消费者”问题演变1（只有同步）：一个缓冲区，一个生产者，一个消费者，生产者不断地生产，消费者不断地消费。只有缓冲区为空时生产者才能放产品，只有缓冲区有数据，消费者才能取产品。用PV操作写出相应的代码段。**

由于缓冲区只有一个，因此只要设置empty和full表示缓冲区是否为空和是否为满就够了，不需要额外的信号量来表示缓冲区使用的互斥。

```
Semaphore empty=1,full=0;
main()
{
        Producer();
        Consumer();
}

Producer()
{
    while(true)
    {
    wait(empty);
    write();     //生产者写入数据
    signal(full);
    }
}

Consumer()
{
    while (true)
    {
    wait(full);
    read();           //消费者读出数据
    signal(empty);
    }
}
```

4.**“生产者-消费者”问题演变2（既有同步，也有互斥）：一个缓冲区，多个生产者，多个消费者，生产者不断地生产，消费者不断地消费。只有缓冲区为空时生产者才能放产品，只有缓冲区有数据，消费者才能取产品。用PV操作写出相应的代码段。**

由于有多个生产者和消费者，所以必须要让相同类型的对象之间互斥。

```
Semaphore mutex=1,empty=1,full=0;
main()
{
    Producer();
    Consumer();
}

Producer()
{
    While(true)
    {
        wait(empty);              //一定要先检查是否可以写再进入互斥区
        wait(mutex)
        write();      //生产者写入数据
        signal (mutex)
        signal(full);
    }
}

Consumer()
{
    While(true)
    {
        wait(full);                    //一定要先检查是否可以读再进入互斥区
        wait(mutex)
        read();           //消费者读出数据
        signal (mutex)
        signal(empty);
    }
}
```

5.桌上有一个空盘，最多可以容纳两个水果，每次只能放入或取出一个水果。爸爸专向盘中放苹果，妈妈专向盘中放橘子。儿子专门等吃盘中橘子，女儿专门等吃盘中苹果，请用PV操作实现爸爸、妈妈、儿子、女儿之间的同步与互斥操作。

```
Semaphore mutex=1,empty=1,full=0;                  //只有一个空盘

father()
{
    while(true)
    {
        wait(empty);        //一定要先检查是否可以放苹果再进入互斥区
        wait(mutex)
        放苹果;
        signal(mutex)
        signal(apple);
    }
}

mother()
{
    while (true)
    {
        wait(empty);
```

```
        wait(mutex)
        放橘子;
        signal(mutex)
        signal(orange);
    }
}

son()
{
    while (true)
    {
        wait(orange);
        wait(mutex)
        取橘子;
        signal(mutex)
        signal(empty);
        吃橘子;
    }
}

daughter()
{
    while (true)
    {
        wait(apple);
        wait(mutex)
        取苹果;
        signal(mutex)
        signal(empty);
        吃苹果;
    }
}
```

2.5 第九周

1. Explain the difference between preemptive and nonpreemptive scheduling.

抢占方式（preemptive）：允许调度程序基于某些原则停止正在运行的进程，将处理器重新分配给另一个进程

非抢占方式（Non-preemptive）：一旦处理器分配给进程，让它执行其任务，直到任务执行结束，或者它被阻塞

2.Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.**

Process	Arrival Time	Burst Time
P1	0.0	8
*P*2	0.4	4
*P*3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processesP1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

a.第一个进程从0s进入，8秒完成，第二个进程0.4秒进入，12秒完成，第三个进程1.0s进入，13秒完成

平均周转时间为 （8 + 11.6 + 12） / 3 = 10.53s

b.第一个进程从0s进入，8秒完成，第二个进程0.4秒进入，13秒完成，第三个进程1.0s进入，9秒完成

平均周转时间为 （8 + 12.6 + 8） / 3 = 9.53s

c.第一个进程从0s进入，14秒完成，第二个进程0.4秒进入，6秒完成，第三个进程1.0s进入，2秒完成

平均周转时间为 （14 + 5.6 + 1） / 3 = 6.87s

3.假设要在一台处理机上执行以下作业,且假定这些作业在时刻0以1,2,3,4,5的顺序到达。（题中时间单位均为s)

表 作业执行时间和优先级

作业	执行时间（s）	优先级
1	10	3
2	1	1
3	2	3
4	1	4
5	5	2

（1）给出分别使用FCFS（先来先服务）、RR（时间片轮转算法，时间片是1）、SJF（最短作业优先）以及非抢占优先调度算法（1的优先级最高），这些作业的执行顺序。画出甘特图。

（2） 针对上述每种调度算法，分别给出平均周转时间。

(1)FCFS:

0 -10s	10s - 11s	11s - 13s	13s - 14s	14 - 19s
1	2	3	4	5

RR：

0 - 1s	1 - 2s	2 - 3s	3 - 4s	4 - 5s	5 - 6s	6 - 7s	7 - 8s	8 - 9s	9 - 10s
1	2（F)	3	4（F)	5	1	3（F)	5	1	5
10s - 11s	11 - 12s	12 - 13s	13 - 14s	14 - 15s	15 - 16s	16 - 17s	17 - 18s	18 - 19s	
1	5	1	5（F)	1	1	1	1	1（F)	

SJF：

0 - 1s	1 - 2s	2 - 4s	4 - 9s	9 - 19s
2	4	3	5	1

优先级调度：

0 - 1s	1 - 6s	6 - 16s	16 - 18s	18- 19s
2	5	1	3	4

(2)FCFS：第一个进程从0s进入，10秒完成，第二个进程0s秒进入，11秒完成，第三个进程0s进入，13秒完成,第四个进程0s进入，14秒完成, 第五个进程0s进入，19秒完成

平均周转时间为 （10 + 11 + 13 + 14 + 19） / 5 = 13.4s

RR：第一个进程从0s进入，19秒完成，第二个进程0s秒进入，2秒完成，第三个进程0s进入，7秒完成,第四个进程0s进入，4秒完成, 第五个进程0s进入，14秒完成

平均周转时间为 （19 + 2 + 7 + 4 + 14） / 5 = 9.2s

SJF：第一个进程从16s进入，19秒完成，第二个进程0s秒进入，1秒完成，第三个进程0s进入，18秒完成,第四个进程0s进入，19秒完成, 第五个进程0s进入，6秒完成

平均周转时间为 （16 + 1 + 18 + 19 + 6） / 5 = 12s

4.What (if any) relation holds between the following pairs of algorithm sets?

a. Priority and SJF

b. Multilevel feedback queues and FCFS

c. Priority and FCFS

d. RR and SJF

a. SJF也是一种优先级算法(任务越短优先级越高)

b.多级反馈队列中同一个队列中的任务是FCFS

c.FCFS也是一种优先级算法(任务来的越早优先级越高)

d.SJF和RR都可以让最短任务最先完成(若是任务同时到达)

5. 下面哪些算法会引起饥饿

a.先来先服务

b.最短工作优先调度

c.轮换法调度

d.优先级调度

a会导致饥饿，前面任务过大导致后面任务不能进行，b和c会导致饥饿，长的任务和优先级低的任务可能不能及时响应。

6 Distinguish between PCS and SCS scheduling.

PCS：进程竞争范围，在执行多对一模型和多对多模型的系统上，线程库调度用户级线程到一个有效的 LWP 上运行，CPU的竞争发生在属于同一进程的线程之间。

SCS：系统范围，决定调度哪个内核线程到CPU ，内核采用SCS来进行。

4.1 第十周

1.Similarly, some systems support many types of structures for a file's data, while others simply support a stream of bytes. What are the advantages and disadvantages of each approach?

课本摘抄

support many types of structures：

the support comes from the system; individual applications are not required to provide the support. It can implement the support presumably more efficiently than an application.

It increases the size of the system. Applications that may require different file types other than what is provided by the system may not be able to run on such systems.

support a stream of bytes:

It simplies the operating system support for file systems. It allows applications to define file structures.

It may cost more time.

2.Provide examples of applications that typically access files according to the following methods:

- **Sequential：** 大文件的连续备份
- **Random：** 大部分的应用在磁盘上的读写

3.Give an example of an application that could benefit from operating system support for random access to indexed files.

(Abstract) An application that maintains a database of entries could benefit from such support. For instance, if a program is maintaining a student database, then accesses to the database cannot be modeled by any predetermined access pattern. The access to records are random and locating the records would be more efficient if the operating system were to provide some form of tree-based index.

4. 从安全性、灵活性和实现几个角度说明加密保护和访问控制两种机制的区别。

控制访问: 根据用户的身份和文件信息判断用户有没有权限访问此文件, 这样的话灵活度不够高，没有权限就不能访问，但是也更为安全。

加密保护:形象地说就是让每一个文件都有一个密码锁。有密码才能访问文件。这样的话灵活度比较高，可以在知道密码后不改变自己的身份访问文件，但是密码可能会泄露，相对来说不安全。

5.选择题

(1)逻辑文件的组织形式是由(　　D)决定的.

A.存储介质特性　B.操作系统的管理方式

C.主存容量　　D.用户

(2)物理文件的组织形式是由(　　D　)决定的

A.应用程序　　B.主存容量

C.外存容量　　D.操作系统

4.2 第十一&十二周

1. **一个文件系统中有一个20MB大文件和一个20KB小文件,当分别采用连续、链接、链接索引、二级索引和UNIX System V 分配方案时(每块大小为4096B,每块地址用4B表示)，问:**

（1）各文件系统管理的最大的文件是多少?

（2）每种方案对大、小两文件各需要多少专用块来记录文件的物理地址(说明各块的用途)？

（3）如需要读大文件前面第5.5KB和后面（16M + 5.5KB）信息，则每个方案各需要多少次硬盘I/O操作?

（1）

文件系统	大小
连续	整个磁盘
链接	每个块4092B存储(4B存下一个块地址)，最多有2^{32}块，最大为$2^{20} * 4092B = 16368GB$
链接索引	若有x个索引块，每个索引块有1023个指针， 最多有x块，则$1023x + x = 2^{32}$，解得$x = 2^{22}$， 于是最大为$x * 1023 * 4096B = 16368GB$
二级索引	每块1024个指针，二级就有2^{20}个指针，于是最大为$2^{20} * 4096B = 4GB$
UNIX S V	$40KB+ 4MB+ 4GB+ 4TB$

(2)

文件系统	专用块
连续	FCB中需要记录首块物理块块号，还有文件总块数，不需要专用块
链接	FCB中需要记录首块物理块块号和最后一块物理块块号，每个物理块还有指向下一块的指针
链接索引	①20KB: 20KB只需要5块，因此只需要1块专用物理块 ②20MB: 需要5000块，每个索引块可以存999个，因此需要6块专用物理块
二级索引	①20KB: 一级索引1块，二级索引1块，因此需要2块专用物理块 ②20MB: 一级索引1块，二级索引5块，因此需要6块专用物理块
UNIX S V	①20KB: 小于40KB，直接寻址，不需要专用物理块 ②20MB: 40KB直接寻址不够，一块一级索引4MB不够，还需要1块二级索引表和其下的4块一级索引，总共需要6块专用物理块

(3)

文件系统	次数
连续	可以先计算出所要信息的块数和所在的物理块(连续可算得)，所以需要硬盘I/O操作1次
链接	① 5.5KB/4092 = 1， 需要硬盘I/O操作2次 ② (5.5KB + 16MB)/4092 = 4107， 需要硬盘I/O操作4108次
链接索引	①5.5KB记录在第1个索引块，需要硬盘I/O操作2次②(5.5KB + 16MB)/(4KB*1023) = 4, 记录在第5个索引块，需要硬盘I/O操作6次
二级索引	都是访问二级索引1次，一级索引1次，实际物理地址一次，需要硬盘I/O操作3次
UNIX S V	①5.5KB直接寻址，需要硬盘I/O操作1次 ②(16MB+5.5KB)在二级索引区域，所以需要硬盘I/O操作3次

2.Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the free-space management scheme in order to support this feature?

做法：在操作系统分配空间的时候，可以尽量先分配 4KB 大小的块(大粒度)，然后剩余的部分分配 512 byte的，这样可以减少分配的块数(与只512 byte相比)，同时减少内部碎片(与只4KB相比)

改变：若是某个文件释放空间，则要检查其前后连续空间是否可以组成 4KB，若是可以的话要组成新的4KB空闲区，否则的话则以 512 byte 存储。

3.Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can bestored in this file system?

一个块里面最多可以存储2048个指针

前12个： 8KB * 12 = 96KB

一级： 2048 × 8KB = 16MB

二级： 2048 × 2048 × 8KB = 32GB

三级： 2048 × 2048 × 2048 × 8KB = 64TB

加起来大约是64TB的大小

5 第十三&十四周

1 State three advantages of placing functionality in a device controller, rather than in the kernel. State three disadvantages.

优点：进行分离之后可以简化内核的算法设计，使得这方面所引起的错误不会导致操作系统的崩溃。

缺点：①bug更难以修复，可能需要新的硬件版本

②想要提高算法则可能需要提高硬件，而不仅仅只是内核和设备控制器的更新

③算法被内嵌被固化，可能与用户程序的使用不一样，造成机器执行效率降低

2.What are the various kinds of performance overhead associated with servicing an interrupt?

①保存上文，保存上文存储的进程信息，当进程重新开始时需要初始化硬件

②刷新CPU中的流水线，当进程重新开始其中的指令需要重新进入流水线

3、Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is:

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

Starting from the current head position, what is the total distance (in

cylinders) that the disk arm moves to satisfy all the pending requests

for each of the following disk-scheduling algorithms?

a. FCFS

b. SSTF

c. SCAN

d. LOOK

e. C-SCAN

f. C-LOOK

注意（The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1805.）磁头运动有方向的

算法	扫描序列	花费
FCFS	2150, 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681	13011
SSTF	2150, 2069, 2296, 2800, 3681, 4965, 1618, 1523, 1212, 544, 356	7586
SCAN	2150, 2296, 2800, 3681, 4965(到4999然后折回), 2069, 1618, 1523, 1212, 544, 356	7492
LOOK	2150, 2296, 2800, 3681, 4965(直接折回), 2069, 1618, 1523, 1212, 544, 356	7424
C-SCAN	2150, 2296, 2800, 3681, 4965(先到4999然后折回到0), 356, 544, 1212, 1523, 1618, 2069	9917
C-LOOK	2150, 2296, 2800, 3681, 4965(直接折回到356), 356, 544, 1212, 1523, 1618, 2069	9137

4、假定磁盘的磁臂现在处于第10柱面，由外向内运动（柱面号由小到大）。现有一组磁盘请求以60、8、15、4、20、40柱面的次序到达磁盘驱动器，磁臂移动一个柱面需要6ms，请完成下面的问题：

（1）访问磁盘所需的时间由哪几部分构成?

（2）采用SSTF算法和SCAN算法进行磁盘调度，请分别给出柱面访问序列，计算平均寻道时间。

(1)①寻道时间：启动磁臂的时间 + 磁头移动到指定磁道的时间

②旋转延迟时间：所需要扇区移动到磁头下面所需要的时间

③传输时间：把数据从磁盘读出或者写入磁盘的时间

(2)SSTF：序列为 10→ 8 → 4 → 15 → 20 → 40 → 60, 共移动了62个柱面，则平均寻道时间为 62 * 6 / 6 = 62ms

SCAN：由于先要往内(柱面号大的方向移动)， 但是并不知道总共有多少个柱面，所以并不清楚那一端最后要到哪里，所以这里先不妨假设最内端即为60

序列为 10 → 15 → 20 → 40 → 60 → 8 → 4, 共移动了106个柱面，则平均寻道时间为 106 * 6 / 6 = 106ms