CS240 Final Exam 2019-Summer | 历年卷解析系列 5-1

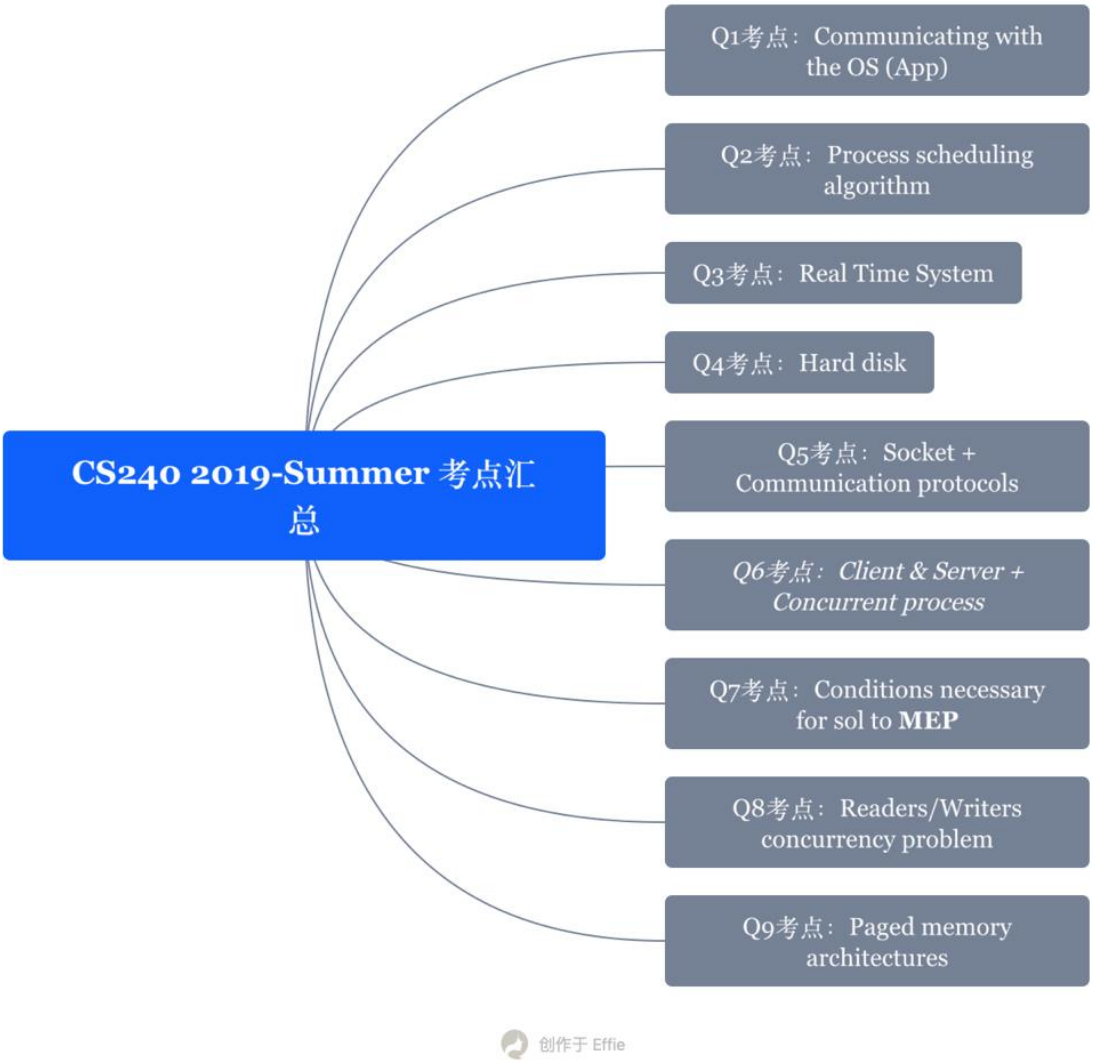
Author : Lance Cai （ HANLIN.CAI.2021@MUMAIL.IE **）**

Update in 2022/06/27

声明：

CS240 2019-Summer 考点汇总

- Q1 考点：Communicating with the OS (App)
- Q2 考点：Process scheduling algorithm
- Q3 考点：Real Time System
- Q4 考点：Hard disk
- Q5 考点：Socket + Communication protocols
- *Q6 考点：Client & Server + Concurrent process*
- Q7 考点：Conditions necessary for sol to **MEP**
- Q8 考点：Readers/Writers concurrency problem
- Q9 考点：Paged memory architectures



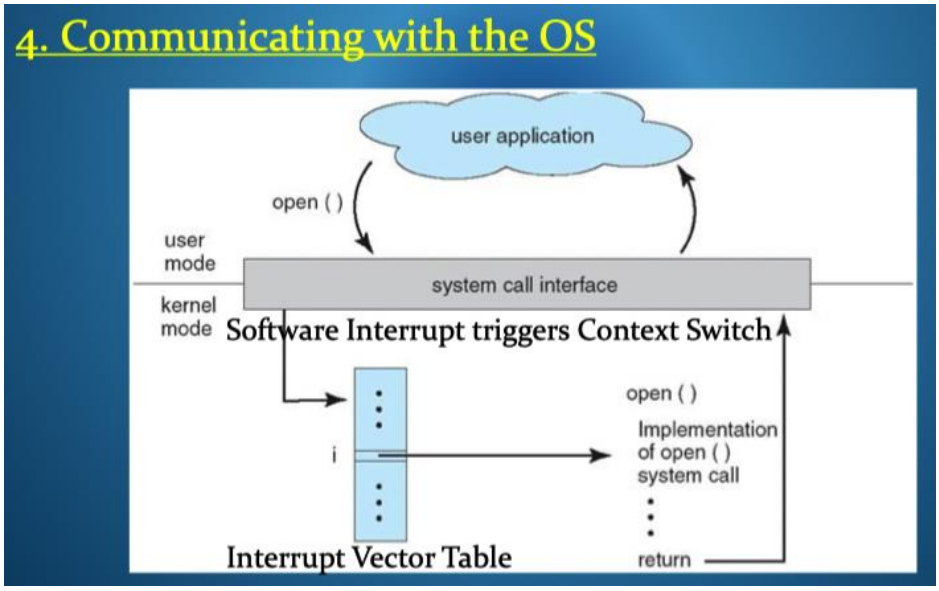
2019-S Q1

> ***Draw a diagram and give a corresponding explanation of how an application process can communicate with an operating system to use system services.***
>
> ***绘制一个应用程序进程如何与操作系统通信以使用系统服务的图表并给出相应的解释。***

考点：Communicating with the OS (App)

课件：L2 Process Management（整合 P45）

1. diagram



2. explanation

应用程序(App)通过与用户状态(User mode)下的库函数(Wrapper function)，访问系统调用(System call)的 API 接口(API interface)，从而能够进入内核状态(Kernel mode)来与操作系统(OS)进行通信，从而使用系统服务(OS services)

The App accesses the API interface of the System call through the Wrapper function in User mode. Hence, it can enter the Kernel mode to communicate with the operating system (OS) and use the OS services.

2019-S Q2

> ***In the context of non-preemptive scheduling, FCFS is an inherently fair algorithm, but may give poor response time. SJF gives optimal response time but may postpone longer tasks indefinitely. Using an example, describe an approach that generally gives better response time than FCFS but does not starve longer jobs.***
>
> ***在非抢占式调度的情况下，FCFS 是一种内在公平的算法，但可能会给出较差的响应时间。SJF 给出最优响应时间，但可能无限期地推迟较长的任务。请举一个例子，描述一种方法，它通常比 FCFS 提供更好的响应时间，但不会饿死更长时间的工作。***

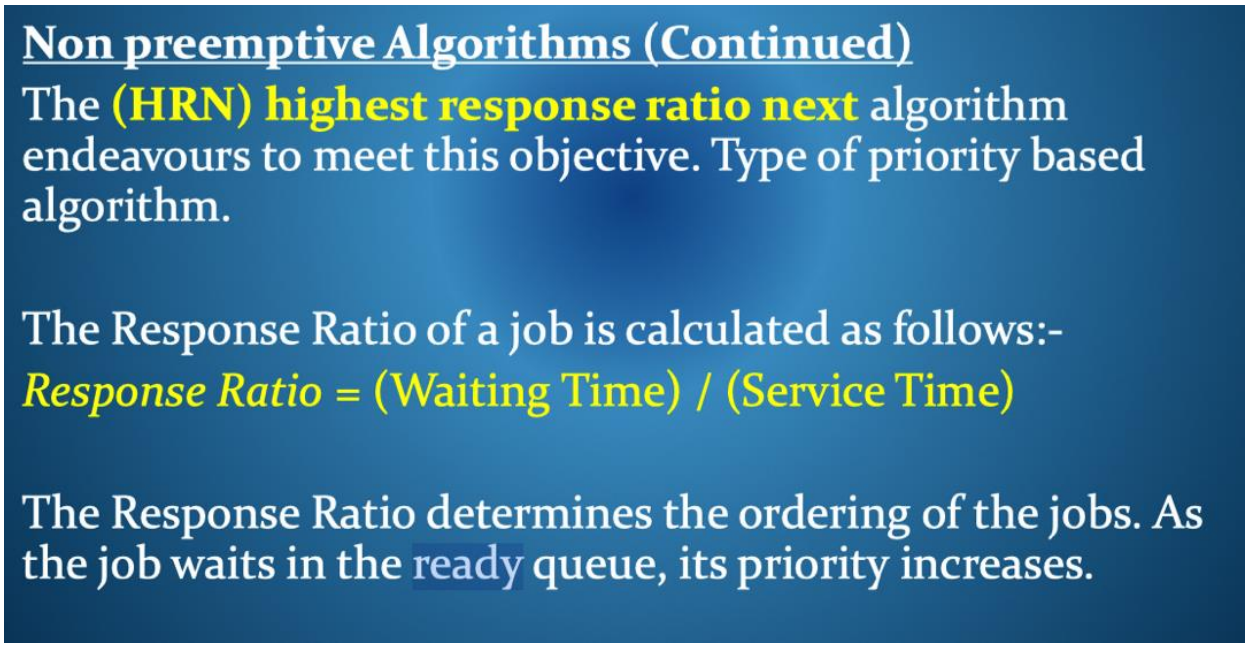
考点：Process scheduling algorithm

课件：The (HRN) highest response ratio next algorithm (整合 P75)

HRRN (Highest Response Ratio Next)
It's a type of *priority based algorithm.*

The longer a job(process) is in the system waiting for the CPU the greater chance it has of being scheduled.

Response Ratio = (wait time) / (service time)



高响应比优先（HRRN, Highest Response Ratio Next）

HRRN

- 算法思想：要综合考虑作业/进程的等待时间和要求服务的时间
- 算法规则：在每次调度时先计算各个作业/进程的响应比，选择响应比最高的作业/进程为其服务

$$响应比 = \frac{等待时间+要求服务时间}{要求服务时间}$$

响应比 ≥ 1

- 用于作业/进程调度：即可用于作业调度，也可用于进程调度
- 是否可抢占？非抢占式的算法。因此只有当前运行的作业/进程主动放弃处理机时，才需要调度，才需要计算响应比
- 优缺点：综合考虑了等待时间和运行时间（要求服务时间）
  等待时间相同时，要求服务时间短的优先（SJF 的优点）
  要求服务时间相同时，等待时间长的优先（FCFS 的优点）
  对于长作业来说，随着等待时间越来越久，其响应比也会越来越大，从而避免了长作业饥饿的问题
- 是否会导致饥饿：不会

知识回顾与重要考点

算法	思想&规则	可抢占？	优点	缺点	考虑到等待时间&运行时间？	会导致饥饿？
FCFS	自己回忆	非抢占式	公平；实现简单	对短作业不利	等待时间√ 运行时间×	不会
SJF/SPF	自己回忆	默认为非抢占式，也有SJF的抢占式版本最短剩余时间优先算法（SRTN）	“最短的”平均等待/周转时间；	对长作业不利，可能导致饥饿；难以做到真正的短作业优先	等待时间× 运行时间√	会
HRRN	自己回忆	非抢占式	上述两种算法的权衡折中，综合考虑的等待时间和运行时间		等待时间√ 运行时间√	不会

注：这几种算法主要关心对用户的公平性、平均周转时间、平均等待时间等评价系统整体性能的指标，但是不关心“响应时间”，也并不区分任务的紧急程度，因此对于用户来说，交互性很糟糕。因此这三种算法一般适合用于早期的批处理系统，当然，FCFS算法也常结合其他的算法使用，在现在也扮演着很重要的角色。而适合用于交互式系统的调度算法将在下个小节介绍...

2019-S Q3

> ***What is a Real Time System? Distinguish between the terms Hard Real Time System and Soft Real Time System in your answer. What approaches might a general purpose operating system take to meet real time process requirements?***

> ***什么是实时系统? 在你的回答中区分术语硬实时系统和软实时系统? 通用操作系统可能采取什么方法来满足实时处理需求?***

考点:

1. Real Time System
2. Hard Real Time System & Soft Real Time System
3. The approaches to meet the require

课件：L5 Multiprocessor and other Systems (整合 P121)

A real time system has well defined, fixed time constraints. Processing must be done within those constraints, if the system break those rules, the hard-real-time system will fail, while the soft-real-time system sometimes can accept occasional violation (sometime can break the rules).

The main features of real-time operating system are time-liness and reliability (Robotics). The approaches is that the OS should base on the priority algorithm and the system call must be preemptive.

我认为以上这里即可，后续为补充：

One of the key things when priority based approaches are used it that the dispatch latency(分派延迟) must be small to ensure that a real time process can start executing quickly. This requires that system calls be preemptible(可抢占式的). It must be possible to preempt the operating system itself if a higher priority real time process want to run.

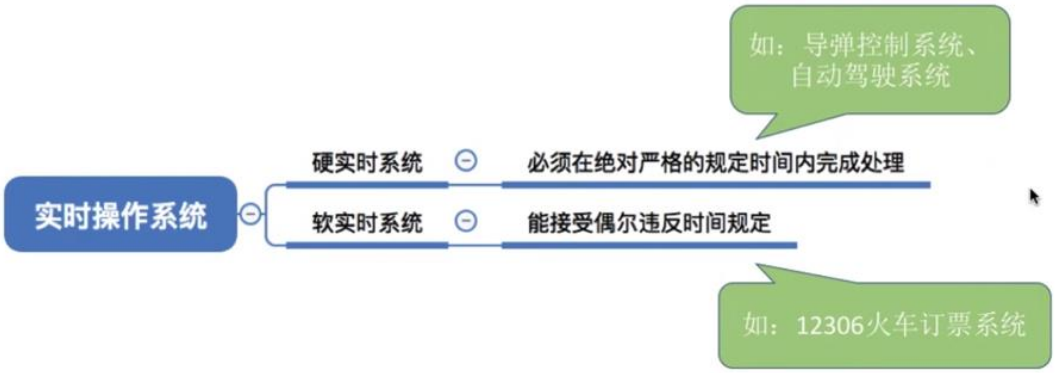
机翻：实时操作系统具有明确的、固定的时间限制，进程任务必须在严格的限制条件下完成，否则系统可能会失败。实时操作系统的主要特点是及时性和可靠性。当使用基于优先级的方法时，关键

的一点是分派延迟必须很小，以确保实时进程可以快速开始执行。这要求系统调用是可抢占的。如果要运行一个更高优先级的实时进程，则必须能够抢占操作系统本身。

实时操作系统

实时操作系统：
主要优点：能够优先响应一些紧急任务，某些紧急任务不需时间片排队。

在实时操作系统的控制下，计算机系统接收到外部信号后及时进行处理，并且要在严格的时限内处理完事件。实时操作系统的主要特点是及时性和可靠性



2019-S Q4

Discuss how data is organised and stored on a hard disk.

讨论数据是如何组织和存储在硬盘上的。

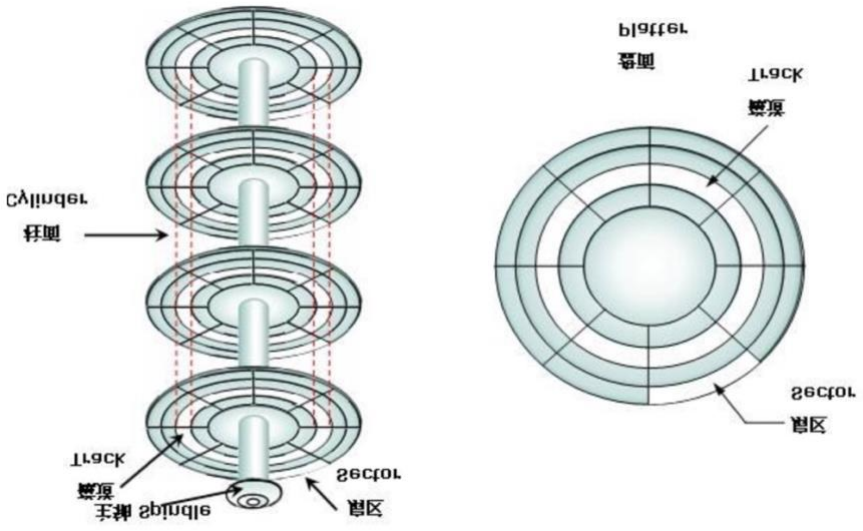
考点：Hard disk

定位: L6 Disk Operation (下述答案为个人总结)

A hard disk is composed of many platters, and the way to store information is to store data through magnetic substances on the surface of the platters. The hard disk can store information representing words, pictures, etc. in binary form.

The surface of the platter is uneven, the convex place is magnetized, representing the number 1, and the concave place is not magnetized, representing the number 0.

机翻：硬盘是由很多的盘片组成，而其存储信息的方式就是通过盘片表面的磁性物质来存储数据。硬盘可以通过二进制的形式来存储表示文字、图片等的信息。盘片表面是凹凸不平的，凸起的地方被磁化，代表数字 1，凹的地方没有被磁化，代表数字 0。



2019-S Q5 难度大

> ***In the context of network communication, what is a socket and how is it used for interprocess communication? Explain the difference between reliable and unreliable communication protocols.***
>
> ***在网络通信的背景中，什么是套接字，以及它如何用于进程间通信?***
>
> ***并解释可靠和不可靠通信协议之间的区别。***

考点：Socket + Communication protocols

定位: L8 Unix Pipes and Sockets (下述为个人总结)

A socket is a data structure created by an application to represent a communication channel. The socket must be bound or associated to some entity (service access point/port) within the communication system. By using communication primitives of the socket interface, the process then exchanges this data from its address space to the address space of the communication subsystem which handles delivery.

Difference:

1. Reliable communication protocols (RCP) is connection oriented, the Unreliable communication protocols(UCP) is connectionless.
2. RCP is transmission data order, UCP does not guarantee the order of data.
3. RCP does not save data boundaries, while UCP does.
4. RCP is slower than UCP, but RCP is more safer.
5. RCP is a heavyweight protocol and UCP is a lightweight protocol.

备注：这里仅作简写，方便排版查看，为了防止歧义，不建议大家写 RCP 和 UCP，

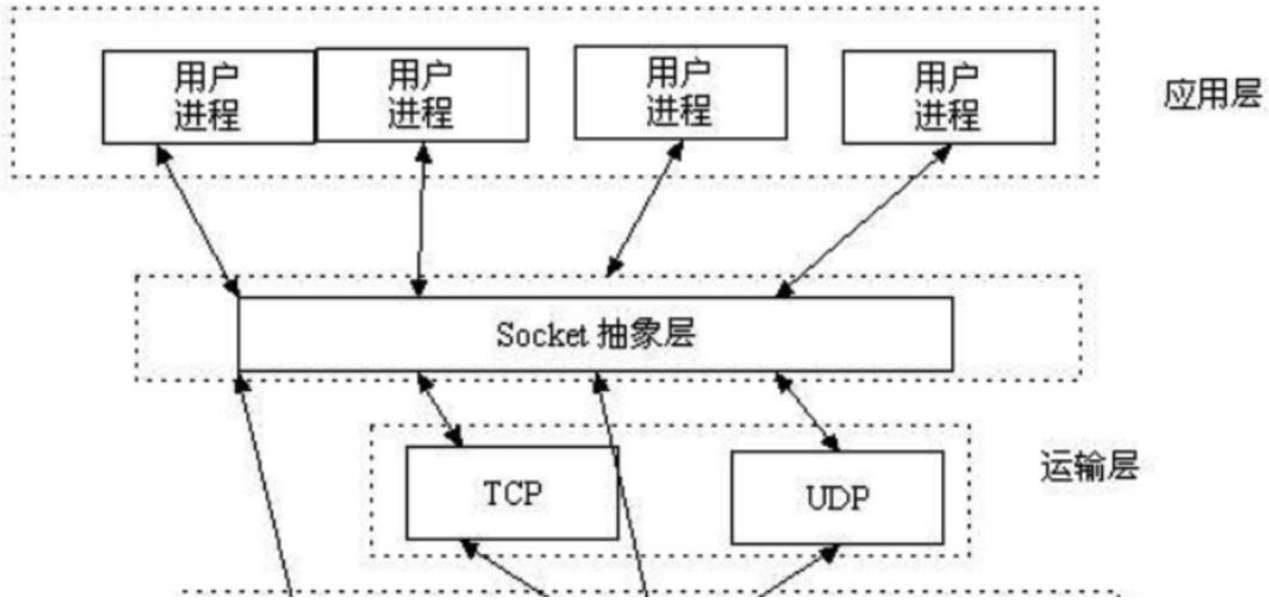
本质上，RCP 对应 TCP，而 UCP 对应的是 UDP，理解万岁。

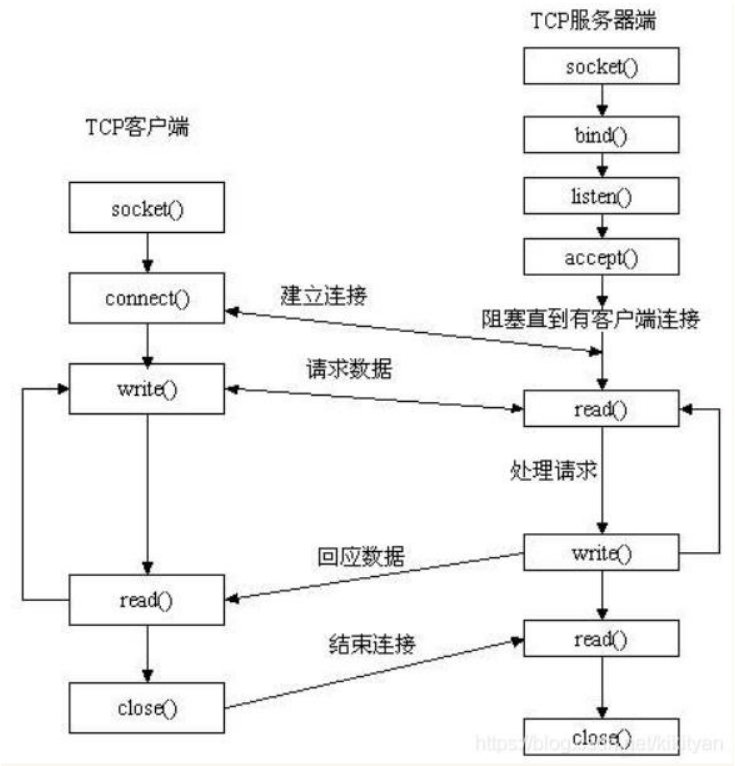
套接字是应用程序创建的数据结构，用于表示通信通道。套接字必须与通信系统中的某些实体(服务接入点/端口)绑定或关联。通过使用套接字接口的通信原语，然后进程将这些数据从它的地址空间交换到处理传递的通信子系统的地址空间。

可靠和不可靠通信协议之间的区别（TCP 和 UDP 协议的区别）：

1. TCP 是可靠传输,UDP 是不可靠传输;
2. TCP 面向连接,UDP 无连接;
3. TCP 传输数据有序,UDP 不保证数据的有序性;
4. TCP 不保存数据边界,UDP 保留数据边界;
5. TCP 传输速度相对 UDP 较慢;

Socket 工作原理：





先从服务器端说起。服务器端先初始化Socket()，然后与端口绑定(bind)，对端口进行监听(listen)，调用accept()阻塞，等待客户端连接。

在这时如果有个客户端初始化一个Socket()，然后连接服务器connect()，如果连接成功，这时客户端与服务器端的连接就建立了。

客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，最后关闭连接，一次交互结束。

Lance 评论：TCP/IP 可以说是必备的知识之一，难但重要，虽然不建议大家在考试时选做这道题，但建议大家务必弄懂 TCP/IP 的底层原理（三次挥手、四次握手等）

参考：CSDN《Socket 通信原理》

http://t.csdn.cn/wC5SO

2019-S Q6

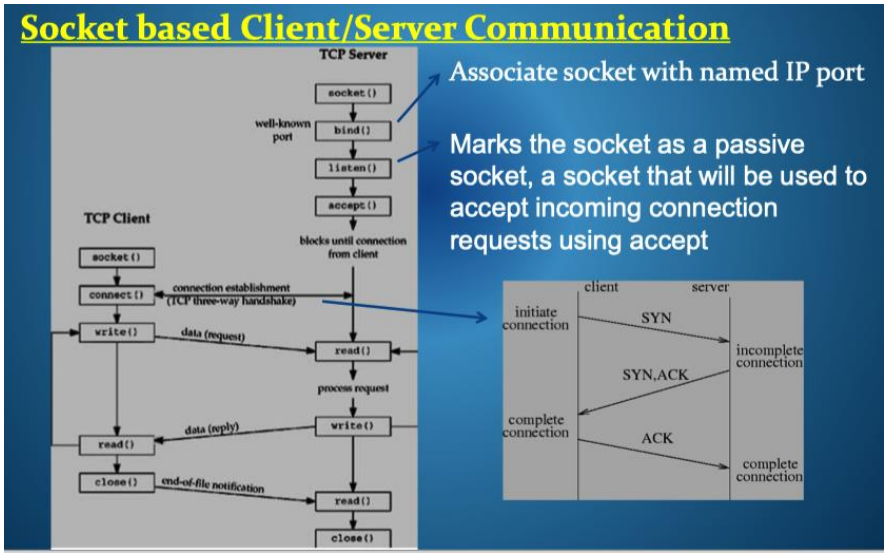
> ***Discuss the general benefits of concurrent threads operating within a client or server application.***
>
> ***讨论在客户机服务器应用程序中，采用并发线程的好处。***

考点：Client & Server + Concurrent process

定位: L8 Unix Pipes and Sockets（以下为个人总结）

1. The use of concurrent programming can realize real-time interaction between the client and the server;
2. It can ensure the correctness of the data;
3. It can ensure that the data sequence is not disrupted.

1. 采用并发编程可以实现客户端和服务器端之间的实时交互；
2. 可以确保数据正确性；
3. 可以保证数据顺序不被打乱；
4. 可靠 Reliable、稳定 Stable、鲁棒性强 Robust；
5. *有时间就再答写套话。*



参考：CSDN《TCP 和 UDP 的优缺点及区别》 http://t.csdn.cn/j7C8I

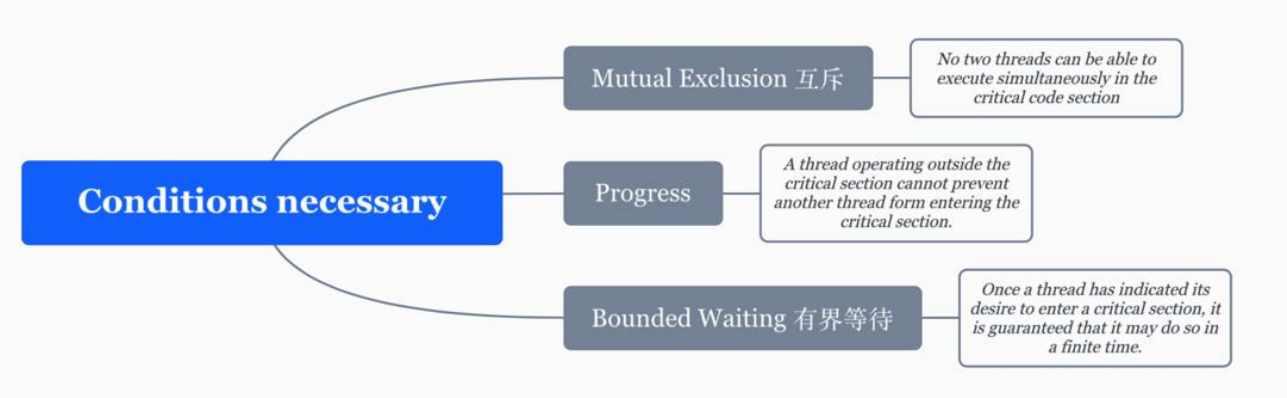
2019-S Q7

State the necessary conditions for a good solution to the mutual exclusion problem.

给出互斥问题良好解的必要条件。

考点：Conditions necessary for sol to **MEP**

定位: L12 Algorithmic Solutions to the Mutual Exclusion Problem



1. No two threads can be able to execute simultaneously in the critical code section
2. A thread operating outside the critical section cannot prevent another thread form entering the critical section.
3. Once a thread has indicated its desire to enter a critical section, it is guaranteed that it may do so in a finite time.

1. 在关键代码段中，不能有两个线程同时执行
2. 在临界区之外运行的线程不能阻止其他线程进入临界区。
3. 一旦线程表示想要进入临界区，它就可以保证在有限的时间内进入临界区。

An outline of the bakery algorithm given on the next page solves the mutual exclusion problem for n processes. Can you identify any inefficiency in the algorithm in terms of its effect on system performance?

下页给出的面包房算法解决了n 个进程的互斥问题。你能从算法对系统性能的影响上找出它的低效率吗?

Question 7 continued:-

The Bakery Algorithm

```
/* Thread i - Choose a ticket */

  choosing[i] = true;
  number[i]=Max(number[0],number[1], ...,number[n-1])+1;
  choosing[i] = false;

/* Wait until we have the lowest ticket */
for (j = 0; j<n; j++) {
    while choosing[j]
        {}

    while ((number[j] != 0) &&  ((number[j],j) < (number[i],i)))  {
        /* Wait if a smaller ticket exists */
        }
}
```

In the program of the bakery algorithm, "while choosing[j]" is executed to perform the loop operation, the process needs to wait in a loop until the lottery ends.

Although mutual exclusion can be achieved, the system continues to occupy CPU resources at this time, resulting in inefficiency.

面包坊算法的程序中，执行 while choosing[j]进行循环操作，进程在这里需要一直循环等待，直到摇号结束。虽然可以实现互斥，但此时系统持续占用 CPU 资源，造成了低效。

补充：在面包坊算法的程序中，没有申请临界区资源的其他进程，是没有必要知道申请了临界区资源的进程是否释放了临界区资源的，因此程序应该等到 Release()操作执行之后，再进行 Reply()操作。

一定要理解面包房算法的原则：

定义(理解这个非常关键！！！)

- 数组choosing[i]为真，表示进程i正在获取它的排队登记号；
- 数组number[i]的值，是进程i的当前排队登记号。如果值为0，表示进程i未参加排队，不想获得该资源。规定这个数组元素的取值没有上界。
- 正在访问临界区的进程如果失败，规定它进入非临界区，number[i]的值置0，即不影响其它进程访问这个互斥资源。

参考：《lamport 面包店算法详细讲解及代码实现》 http://t.csdn.cn/7uFQw

2019-S Q8

Outline what is meant by the Readers/Writers concurrency problem and outline a solution which prioritises readers.

概述什么是读者/作者并发性问题，并概述一个优先考虑读者的解决方案。

考点：Readers/Writers concurrency problem

定位: L12 Readers/Writers problem (下述答案为个人总结)

There are two concurrent（并发） processes of readers and writers, sharing a file, in which the reader processes can occur in parallel（并行） without affecting each other. But the writer process must have exclusive access (exclusively the shared resource to ensure no data inconsistency errors), which is the reader-writer problem.

撰写：有读者和写者两组并发进程，共享一个文件，其中读者进程可以并行发生，不会相互影响。但写者进程必须独占访问（独占共享资源，以确保不会发生数据不一致的错误），这就是读者-写者问题。

1. Readers and writers are mutually exclusive.We will use a semaphore (wrt) to control this.
2. A writer must acquire the wrt semaphore before writing, and the first reader must acquire the wrt semaphore to prevent writers.
3. If a reader is finished with the data, and there are no other readers using the item, then the last reader can give up the semaphore (wrt) to allow any waiting writers to enter
4. In order to determine if a reader is the first or last reader, we will have to keep a count (readerCount) of the number of readers accessing the item at one time.This variable will be incremented as new readers access the data and decremented as new readers finish with the data.
5. We will use a semaphore mutex to ensure that only one reader thread at a time tests and modifies readerCount.

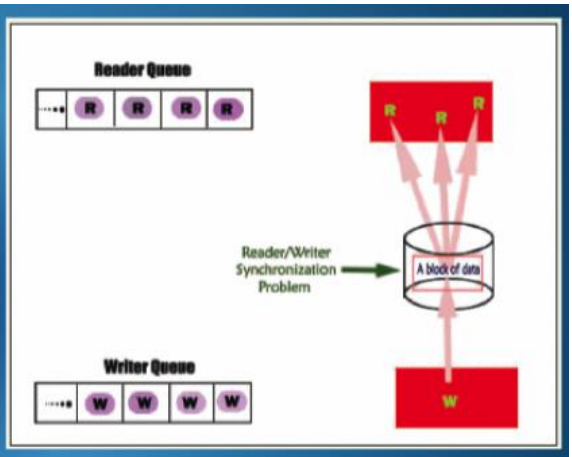
互斥 信号量 wrt，初值是1，代表一个共享文件，解决“读－写”互斥，“写－写”互斥。

一个记数器，即 整型 变量readcount，记录读者数，初值是0。 来一个读者， readcount加1 当readcount ＝1表示是第一个读者， 则需要执行p操作抢占文件；否则表示已有读者在安全的读数据。 走一个读者，readcount减1 当readcount ＝0表示是最后一个读者，则需要v操作释放资源；否则表示还有读者在读数据。

readcount 为多个读者共享的变量，是临界资源。用互斥信号量mutex控制, mutex初值是1。

Readers/Writers Problem

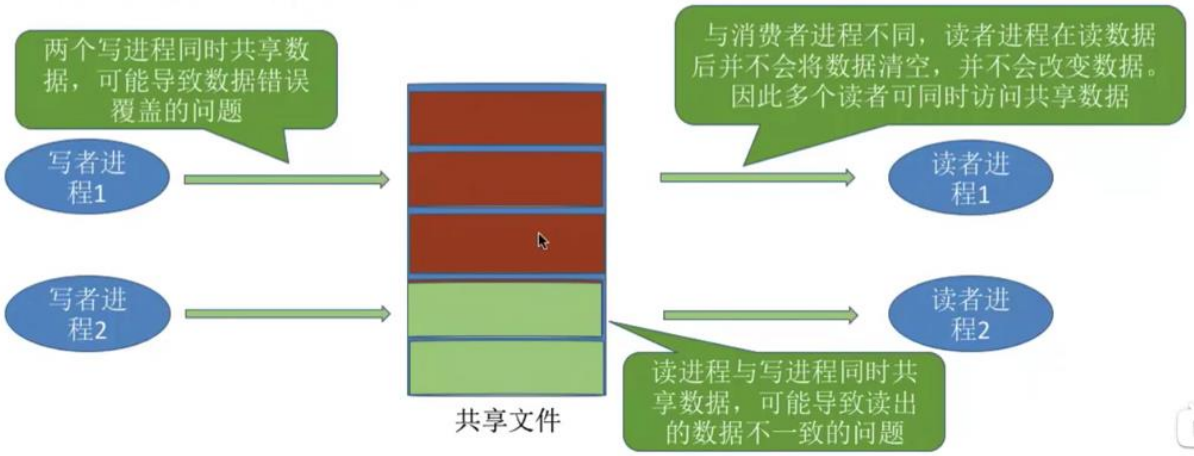
A number of concurrent threads wish to carry out operations on a shared data. Some operations may involve reading the data and others involve writing/updating it.



While reader threads can access the data concurrently without any adverse affects, a writer thread requires exclusive access to carry out its transaction.

The readers/writers problem has a number of alternative solutions, which try to address efficiency and fairness to reader and/or writer threads in different ways.

有读者和写者两组并发进程，共享一个文件，当两个或两个以上的读进程同时访问共享数据时不会产生副作用，但若某个写进程和其他进程（读进程或写进程）同时访问共享数据时则可能导致数据不一致的错误。因此要求：①允许多个读者可以同时对文件执行读操作；②只允许一个写者往文件中写信息；③任一写者在完成写操作之前不允许其他读者或写者工作；④写者执行写操作前，应让已有的读者和写者全部退出。



并发 Concurrent & 并行 Parallel 的区别

1. 读进程(Reader)优先的策略 代码实现：

```
semaphore rw=1;      //用于实现对文件的互斥访问。表示当前是否有进程在访问共享文件
int count = 0;       //记录当前有几个读进程在访问文件
semaphore mutex = 1;//用于保证对count变量的互斥访问
```

```
writer (){
   while(1){
       P(rw);    //写之前"加锁"
       写文件...
       V(rw);    //写之后"解锁"
   }
}
```

思考：若两个读进程并发执行，则两个读进程有可能先后执行 P(rw)，从而使第二个读进程阻塞的情况。
如何解决：出现上述问题的原因在于对 count 变量的检查和赋值无法一气呵成，因此可以设置另一个互斥信号量来保证各读进程对count 的访问是互斥的。

潜在的问题：只要有读进程还在读，写进程就要一直阻塞等待，可能"饿死"。因此，这种算法中，读进程是优先的

```
reader (){
   while(1){
       P(mutex);      //各读进程互斥访问count
       if(count==0)
           P(rw);     //第一个读进程负责"加锁"
       count++;       //访问文件的读进程数+1
       V(mutex);
       读文件...
       P(mutex);      //各读进程互斥访问count
       count--;       //访问文件的读进程数-1
       if(count==0)
           V(rw);     //最后一个读进程负责"解锁"
       V(mutex);
   }
}
```

在读者-写者问题中：

2 个互斥关系，1 个并行关系

1、读者-写者是需要互斥的 2、写者-写着是需要互斥的 3、读者-读者是可以并行的

核心思想在于设置一个 count 来记录当前访问共享文件(临界区)的读者进程数量。

用 count 来判断是否是第一个/最后一个读进程：

1、第一个，进行加锁 2、最后个，读完之后解锁

2019-S Q9

> ***Explain the advantages of using paged memory architectures. Are there any drawbacks to using this approach?***
>
> ***解释使用分页内存架构的优点。使用这种方法有什么缺点吗?***

考点：Paged memory architectures

定位：Method 4 - Paged Architecture（整合 P456）

Advantages：

1. No outer fragments, the last page may have inner fragments but not big;
2. The architectures does not have to be stored continuously;
3. It is convenient to change the size of the space occupied.

Disadvantages：

1. The program needs to be fully loaded into memory, which increased machine cost and system overhead.

优点：

1 没有外部碎片，最后一页可能有内碎片但不大;

2 程序不必连续存放;

3 便于改变程序占用空间大小。

缺点：

1 程序仍需要全部装入内存（基本页式存储管理）

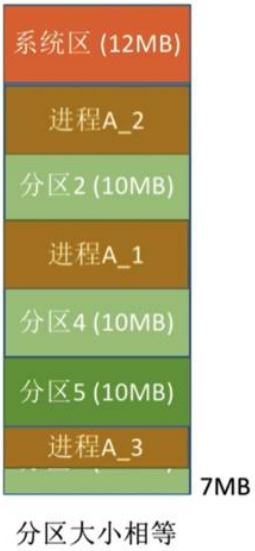
如：地址变换机构缺页中断的产生和选择淘汰页面等，增加了机器成本和系统开销。

基本分页存储管理的思想:

假设进程A大小为 23MB，但是每个分区大小只有 10MB，如果进程只能占用一个分区，那显然放不下。
解决思路：如果允许进程占用多个分区，那么可以把进程拆分成 10MB+10MB+3MB 三个部分，再把这三个部分分别放到三个分区中（这些分区不要求连续）...

进程A 的最后一个部分是 3MB，放入分区后会产生 7MB 的内部碎片。
如果每个分区大小为 2MB，那么进程A 可以拆分成 11 * 2MB + 1MB 共 12 个部分，只有最后一部分 1MB 占不满分区，会产生 1MB 的内部碎片。
显然，如果把分区大小设置的更小一些，内部碎片会更小，内存利用率会更高。

基本分页存储管理的思想——把内存分为一个个相等的小分区，再按照分区大小把进程拆分成一个个小部分



分区大小相等

将内存空间分为一个个大小相等的分区（比如：每个分区 4KB），每个分区就是一个“页框”，或称“页帧”、“内存块”、“物理块”。每个页框有一个编号，即“页框号”（或者“内存块号”、“页帧号”、“物理块号”）页框号***从0开始***。

将用户进程的地址空间也分为***与页框大小相等***的一个个区域，称为“页”或“页面”。每个页面也有一个编号，即“页号”，页号也是***从0开始***。
（注：进程的最后一个页面可能没有一个页框那么大。因此，页框不能太大，否则可能产生过大的内部碎片）

操作系统以页框为单位为各个进程分配内存空间。进程的每个页面分别放入一个页框中。也就是说，进程的页面与内存的页框有一一对应的关系。
各个页面不必连续存放，也不必按先后顺序来，可以放到不相邻的各个页框中。

参考：《操作系统：连续分配、分页和分段三种存储分配机制的优缺点》

http://t.csdn.cn/FbQo7

Lance 写在最后

如果你发现了材料内容的重大缺陷与错误，非常期待你可以给我 email（邮箱号已贴在最上方）。随信附上修改 Tips——我将不胜感激！

祝你考试顺利，学业有成。 我们高处见。

如果你想找到我: www.mieclance.club