OS 期末考试复习通宵版

第一章 6 分

1. 计算机系统的定义有哪几个组成部分？

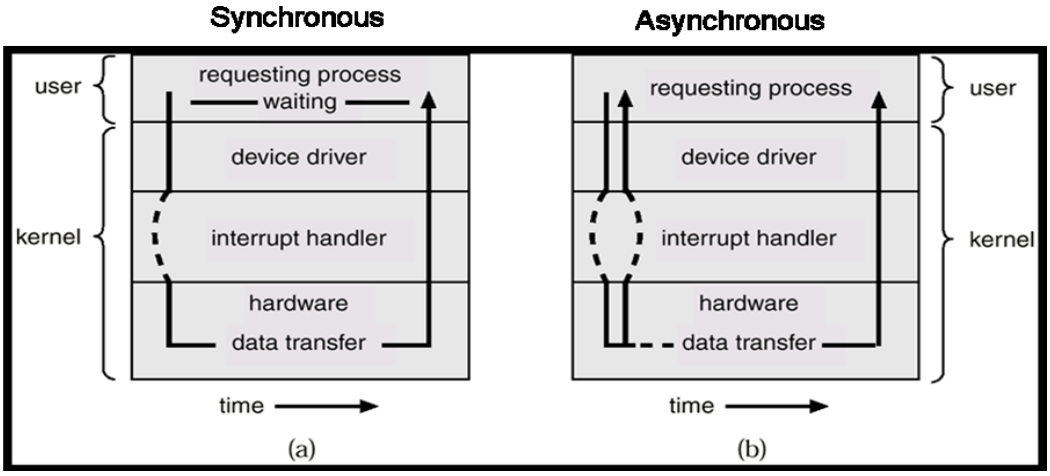
答： A computer system can be divided roughly into four components: the hardware, the operating system, the application programs and the users; (Figure 1.1 of chapter 1);

2. 操作系统功能两横两纵的描述？

答：We can view the operating system from two viewpoints, that of the user and that of the system. (User view and System view). Operating system is resource allocator and a control program. (温馨提示：此题存在较大争议，该答案仅供参考，如果有精确的解答，期待您的指正)

3. I/O 的基本方式

答：Two I/O Methods, Synchronous and Asynchronous;



4. 存储器层次结构

答：The Storage hierarchy in a top-down approach can be: registers, cache, main memory, electronic disk, magnetic disk, optical disk and magnetic tapes. （See figure 1.4 of chapter 1）

5. 二元模式

答：User mode and Kernel mode (also called supervisor mode, system mode or privileged mode). A bit, called a mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1); the operating system keeps changing from one mode to the other mode ;(You can see the switch of mode in figure 1.8 of chapter 1).

6. 计算环境

答：(1) Traditional Computing
(2) Client-Server Computing
(3) Peer-to-Peer Computing
(4) Web-based Computing

第二章 8 分

1.用户接口类型

答：Command Interpreter; Graphical User Interface.

2.在 system call、API、System Programs 三个层次实现系统调用的联系和区别

答：System calls provide an interface to the services made available by an operation system. API specifies a set of functions that are available to an application programmer, and behind the scene, the functions that make up an API typically invoke the actual system calls on behalf of the application programmer; API concerns program portability and actual system calls can often be

more detailed than it may appear. System programs provide a convenient environment for program development and execution. Some of them are simply user interface to system calls.

(It's really a boring question! Write as more as you can!)

3.操作系统设计和实现时需要考虑的问题

答：(1) Define goals and specification!(Design concerns)

(2) Separate policy from mechanism.(Design concerns)

(3) Choose the programming language to implement the operating system. (Implementation concerns)

(4) Take the performance improvements into consideration; choose better data structures and algorithms. (Implementation concerns)

(5) Monitor system performance. (Implementation concerns)

4.操作系统需要提供的服务和主要模块

答：**Operating-system service：**

UI, Program execution, I/O operations, File-system manipulation, Communications, Error detection.

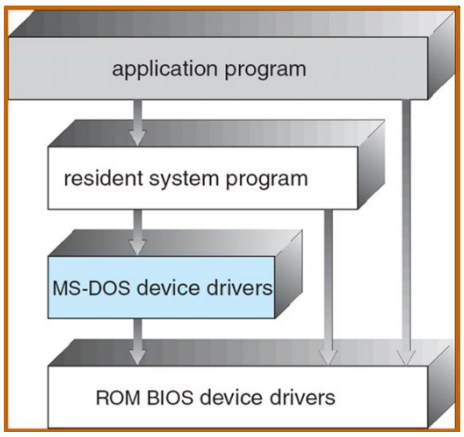
Kernel modules:

Scheduling classes, File systems, Loadable system calls, Executable formats, STREAMS modules, Miscellaneous, Device and bus drivers.

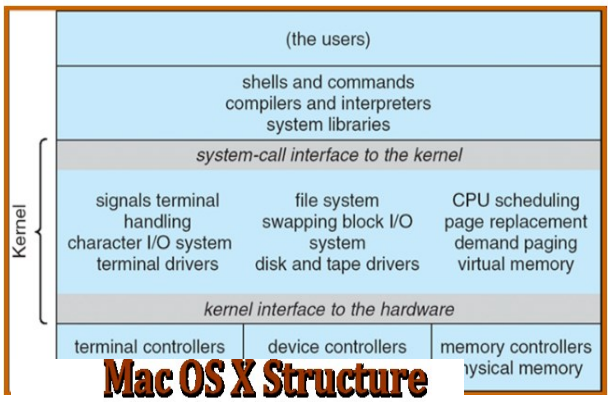
5.主流的操作系统结构

答

MS-DOS Layer Structure

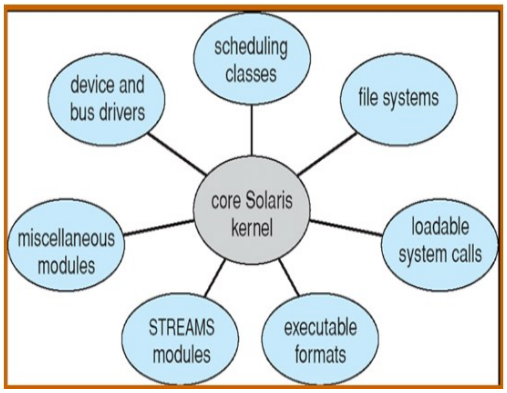


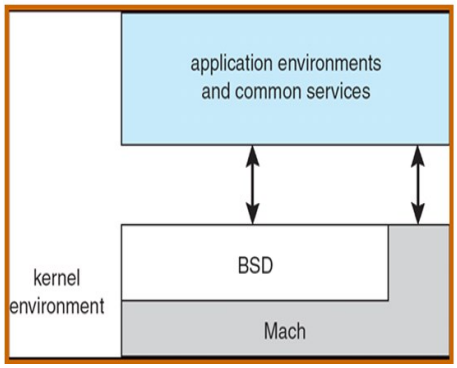
UNIX System Structure



Mac OS X Structure

Solaris Modular Approach





6.操作系统如何加载

答：Load a small piece of code known as the bootstrap program or bootstrap loader into main memory, and starts its execution. The bootstrap program performs a variety of tasks, after the bootstrap program has been loaded, it can traverse the file system to find the operating system kernel, load it into memory, and start its execution. It only at this point that the system is said to be running.

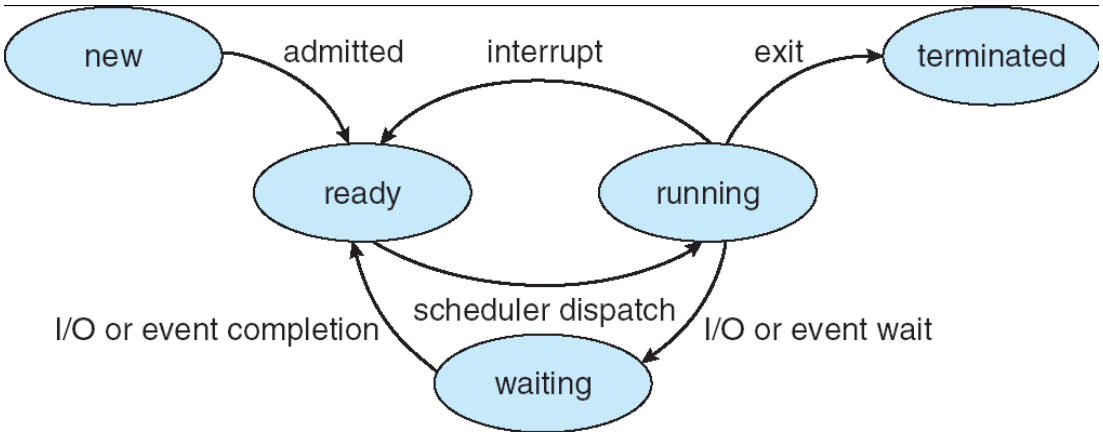
第三章 10 分

1. 进程的概念

答： Process is a program in execution that forms the basis of all computation. A process is the unit of a work in a modern time-sharing system.

2. 掌握进程状态迁移图和进程队列迁移图（包括状态名称和转移条件）

答：



As a process executes, it changes *state*

New: The process is being created

Running: Instructions are being executed

Waiting: The process is waiting for some event to occur(such as an I/O completion or reception of a signal)

Ready: The process is waiting to be assigned to a processor

Terminated: The process has finished execution

3.PCB 应该包含哪些内容

答： Program control block(PCB)

Process state, Program counter, CPU registers

CPU scheduling information, Memory-management information

Accounting information, I/O status information

4.什么叫做长期调度、中期调度、短期调度，区别和联系时什么？

答：**Long-term scheduler** (or job scheduler) – selects which processes should be brought into the ready queue.

Medium-term scheduler – this schema is called swapping. It removes process from memory.

Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU.

The long-term scheduler executes much less frequently; the long-term scheduler controls the degree of multiprogramming. Medium-term scheduler is introduced in time-sharing system.

5.进程为什么按照 I/O 绑定和 CPU 绑定进行分类?

答：I/O and CPU are the main two resources in the computer. I/O-bound process spends more time

doing I/O than computations, many short CPU bursts. CPU-bound process spends more time doing computations; few very long CPU bursts. With careful scheduling of these two kinds of process, we can achieve the highest performance.

6. 进程上下文切换

答：When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.

Context-switch time is overhead; the system does no useful work while switching(这段时间称之为垃圾时间).

Time dependent on hardware support.

7. 进程的创建与结束的内容

答：**Process Creation：**

Parent process create children processes, which, in turn create other processes, forming a tree of processes

Resource sharing

1） Parent and children share all resources

2） Children share subset of parent's resources

3） Parent and child share no resources

Execution

1） Parent and children execute concurrently

2） Parent waits until children terminate

Address space

1） Child duplicate of parent

2） Child has a program loaded into it

Process Termination:

Process executes last statement and asks the operating system to delete it (**exit**)

a) Output data from child to parent (via **wait**)

b) Process' resources are deallocated by operating system

Parent may terminate execution of children processes (**abort**)

a) Child has exceeded allocated resources

b) Task assigned to child is no longer required

c) If parent is exiting.

Some operating system do not allow child to continue if its parent terminates, or all children terminated - *cascading termination*

8. 进程通信的基本模型

答：There are two fundamental models of interpocess communication: (1) shared memory (2) messaging passing

9.进程通信的手段分为哪两个大类，在这两大类中的 SEND 和 Receive 原语会有什么不同？

答：**Direct Communication：**

Send (*P, message*) – send a message to process P

Receive (*Q, message*) – receive a message from process Q

Indirect Communication：

Send (*A, message*) – send a message to mailbox A

Receive (*A, message*) – receive a message from mailbox A

10. 你所了解的进程通信技术有哪些？

答： Pipeline communication (管道通信，分为命名管道和匿名管道)
Message system(消息通信)
Network (网络方式)
Shared memory(共享内存)

11.什么叫做套接字用来干什么？

答:A socket is defined as an endpoint for communication. A socket is indentified by an IP address concatenated with a port number. Communication consists between a pair of sockets

第四章 4 分

1. 线程的概念，实现线程的三种基本模型是什么？

答：A thread is the basic unit of CPU utilization.

Many-to-One, One-to-One, Many-to-Many

2.为什么要采用线程，和进程的区别与联系

答：Benefits: Responsiveness, Resource Sharing, Economy, Utilization of MP Architectures

进程是一个执行中的程序，一个进程中始终有一个主控制线程。一个进程可以创建多个线程，由同一个进程创建的多个线程之间共享代码和数据，无需多个进程间的通信操作。创建一个线程的开销远小于创建一个进程。（温馨提示：多扯点，前提是扯的靠谱）

第五章 10 分

1.什么是 CPU 调度发生在什么时候，什么是 I/O 突发周期和 CPU 突发周期？

答：Maximum CPU utilization obtained with multiprogramming.
CPU-burst: Process is in execution in the CPU
IO-burst: Processing is in I/O Operation

2.CPU 调度可以分为哪两个大类？

答：Preemptive scheduling and nonpreemptive scheduling.

3.处理 CPU 调度的系统模块叫什么，干什么？

答：CPU Scheduler, Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

4.调度的评价标准是哪些？

答：**CPU utilization** – keep the CPU as busy as possible
Throughput – # of processes that complete their execution per time unit
Turnaround time – amount of time to execute a particular process
Waiting time – amount of time a process has been waiting in the ready queue
Response time – amount of time it takes from when a request was submitted until the first response is produced.

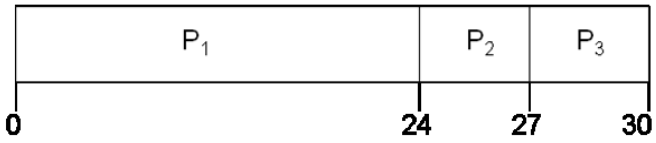
5.FCFS、SJF、RR 的算法思想以及平均等待时间计算，特别注意 SJF 有两种方式。

答：FCFS：最公平的调度思想，谁先来就先服务谁，大家想一想日常生活中的队列，就是这样的一种 FCFS 的典型。

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



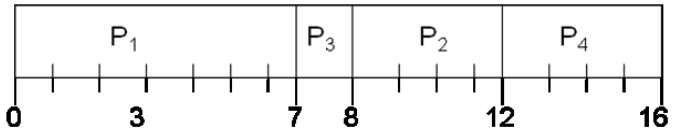
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

SJF：短作业优先，挑软柿子捏，同一批到达的作业，谁的时间短，就先给谁服务，同时分抢占性和非抢占性两种情况。
非抢占性：为您服务，不离不弃。接受到任务后会一致执行到结束。

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



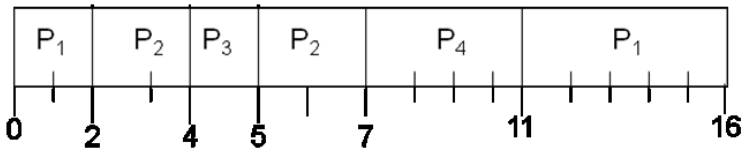
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

抢占性：“见钱眼开”，如果在执行中如果存在待执行的作业的执行时间短于当前作业的剩余时间，当前作业被抢断。

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

注：P2 被抢断，其等待时间应为 5-4=1;注意，等待时间从某次等待的时刻算起，到再次被执行时结束。

RR：时间片乱转，最不得罪人的调度策略。执行人人有份，你方唱罢我登台。注意：如果在一个时间片结束之前，进程已经执行结束，则立即进入下个时间片的调度！（CPU 绝不会干耗完剩余的时间）

6.多级队列和多级反馈队列的思想及各自的优缺点

答：多级队列实际上就是根据进程不同的性质（是前台进程还是后台进程等）划分不同的调度队列，不同的调度队列被赋予了不同的优先级，一般而言采取先服务高优先级的队列再服务低优先级队列的方式，优点当然是提高了前台进程或者说高优先级进程的响应速度，缺点是可能导致饥饿。多优先级反馈队列实际上是在多级队列上增加了一个线程在不同优先级的队列之间的迁移机制（要想富，动干部，呵呵，括号里的话千万别在考试时写！）。优点可以说是改善了死锁，缺点是算法复杂，参数设置较为困难（比如如何划分优先级队列，在什么条件下改变进程的优先级）。

第六章 12 分

注：应热心同学的建议，以下章节采用中英文对照的形式解答，前提是不损害表达的精确性。对无法得到确切翻译的概念，依旧采用英文解答。

1.竞争条件的定义、具体表现的例子？

答：A situation like where several process access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a race condition.

For example, is now count is equal to 5, there are two threads to operate on count, one will be “count++” and the other is “count--”, and we know that “count++” and “count--” both consist of three independent instructions, so the final output of count depends. (书上的例子，大家把那个例子抄下就行了，我的这段话仅作示例)。

2.解决竞争条件的三条准则是什么？

答：1. Mutual Exclusion - If process P_i is executing in its critical section, then no other processes can be executing in their critical sections（互斥）

2. Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely（有空让进）

3. Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted（有限等待）

- Assume that each process executes at a nonzero speed
- No assumption concerning relative speed of the N processes

3.Peterson 算法的思想及证明？

答：Peterson 算法的关键思想就是两把锁，第六版的中文教材上阐述了分别只使用 flag 标记和 turn 变量的时候的错误情形，从而证明了两把锁的必要性。证明参考书上的一大段话即可。

4.在指令集、操作系统内核、编译器三个层次上实现的同步手段分别是什么？

答：指令集层次：提供特殊的硬件指令，使得特定代码段可以顺序执行不被抢占（仅仅适用于单 CPU 的情形）。

操作系统内核层次：操作系统内核分为抢占式内核和非抢占式内核，其中非抢占式内核可免除竞争条件的困扰。

编译器层次：可以通过同步信号量（如 semaphore）等同步原语来控制线程的同步。

5.有限缓冲区、读者作者、哲学家问题分别描述了什么样的进程同步问题？

答：有限缓冲区问题实际上就是生产者—消费者问题，生产者向一个有限的缓冲区中写数据，消费者从中读数据。线程同步需要使得生产者和消费者不能同时操作缓冲区，且消费者必须在生产者向缓冲区中写入了数据之后才能进行读取（教材中的 full 信号量就是这个作用）。

读者作者问题是指有多个线程对某个共享数据进行操作，一部分线程对数据进行写操作，另一部分线程对数据进行读操作，写操作的线程互斥执行，即同时只能有一个写操作线程对数据进行写操作，读操作的线程可以并发对数据进行读取，但只要有一个读操作线程在对数据进行读取，写操作线程就不能对数据进行写操作。

哲学家就餐问题是一个典型的并发控制问题，其关键是解决多个资源在多个线程之间的分配问题，同时要避免死锁和饥饿。

第七章 8 分

1.死锁的概念？

答: In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources, and if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock. (中文答案：多任务环境中，一个线程因为所请求的资源被另外一个处于等待状态的进程占据而陷于等待状态的情形称之为死锁)

2.死锁的必要条件？

答 ：

- **Mutual exclusion:** only one process at a time can use a resource.
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes.
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_0\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by

  $P_2, \ldots, P_{n-1}$ is waiting for a resource that is held by P_n, and P_0 is waiting for a resource that is held by P_0.

中文参考回答：互斥执行，把持并等待，不允许抢占，循环等待

3.死锁的解决方案分为哪两个大类？

答：方式一：Ensure that the system will *never* enter a deadlock state.

方式二：Allow the system to enter a deadlock state and then recover.

（温馨提示：严格意义上说，书上给了第三种解决方案，就是不理会死锁，就当成死锁没有发生，上面给的两种方式之所以成为两类，因为第一类是不让死锁发生，第二类是死锁发生后补救，属于两大类解决方案，至于第三种解决方案，我也无法给出准确的分类，如果有精确的解答，期待您的指正）

4．死锁预防和死锁避免的联系与区别？

答：**Deadlock prevention** provides a set of methods for ensuring that at least one of the necessary conditions cannot hold. These methods prevent deadlocks by constraining how requests for resources can be made.

Deadlock avoidance requires that operating system be given in advance additional information concerning which resources a process will request and use during lifetime. The system allocates the resources according to the additional information.

They both ensure that the system will never enter a deadlock state.

5.死锁的检测与恢复？

答：着重掌握死锁的检测算法；恢复模式(Recovery schema), 分为 Process Termination 和 Resource Preemption.

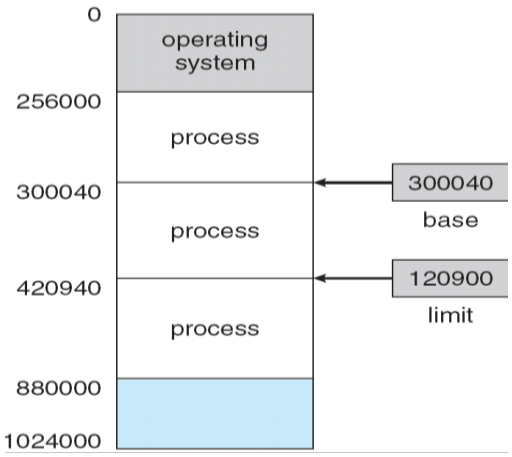
6.操作系统具体会采取哪些措施处理死锁？

答：Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX.（简单的说：不管不问）

第八章 16 分

1.地址转换的基本方式是什么？

答：



Base register and limit register(基址变址法)

2.地址绑定的三个阶段是什么，有什么区别和联系？

答：Address binding of instructions and data to memory addresses can happen at three different stages：

Compile time: If you know at compile time where the process will reside in memory, absolute code can be generated; or it must recompile code if starting location changes.

Load time: Must generate relocatable code if memory location is not known at compile time. In this case, final binding is delayed until load time.

Execution time: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)

(温馨提示：地址绑定就是一种地址空间到另一种地址空间的映射)

3.MMU 是什么会做哪些工作

答：MMU is the acronym of memory-management unit; it is hardware device that maps virtual to physical address. (温馨提示：内存管理单元，实现由虚地址向物理地址映射的硬件单元)

4.连续分配策略的三种选择是什么，连续分配的优缺点？

答：**First-fit**: Allocate the *first* hole that is big enough

Best-fit: Allocate the *smallest* hole that is big enough; must search

Worst-fit: Allocate the *largest* hole; must also search entire list

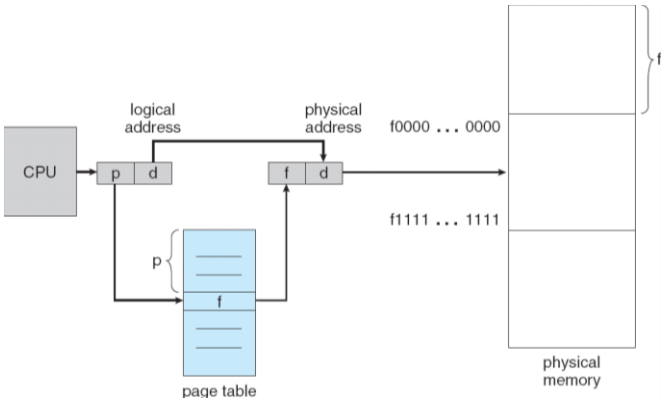
The continuous memory allocation suffers from the external fragmentation(外部碎片)。

5.什么是外部碎片和内部碎片，有什么区别，具体例子？

答：**External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous

Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.（温馨提示：外部碎片即没有被分配的内存空间，而内部碎片是指分配给了一个进程却没有被使用的内存空间）

6.分页方式的内存管理机制是怎样的，在分页方式下怎么进行地址转换？



答：（万般奥妙，尽在图中）

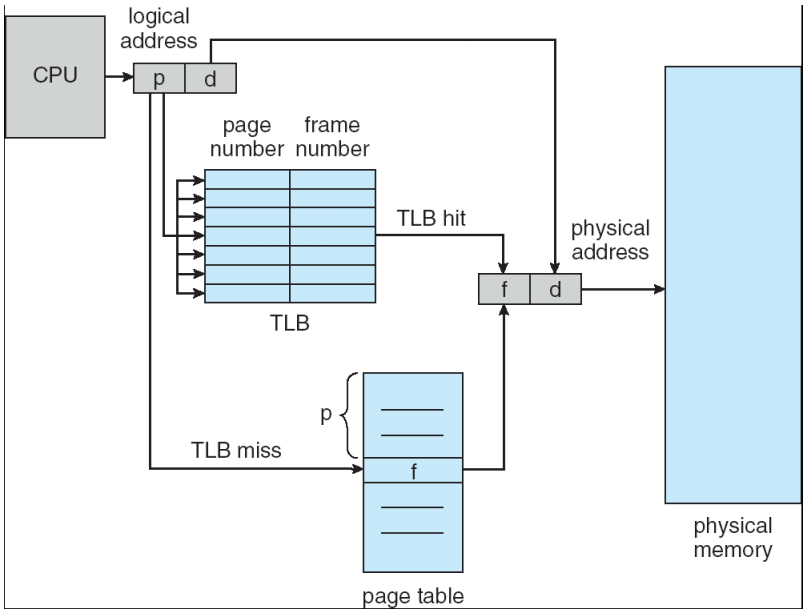
分页是一种允许一个进程的物理地址不连续的内存管理模式，因此分页机制的本质在于提供一种虚地址到物理地址的不连续映射机制,在分页模式下的地址转换实际上就是一个“查表”的过程，将进程中的页(page)与物理内存中的（frame）对应起来)。

7.页表放在哪里有什么内容？

答：页表放在主存中，每个操作系统提供了自己的页表存储机制。大部分操作系统为每一个进程分配一个页表。页表的指针与其他的寄存器值一起被存储在 PCB（process control block）中。

8.TLB 是什么，放在哪里，内容是什么？

答：TLB is the acronym of translation look-aside buffer. The TLB is associative, high-speed memory. TLB contains a few of the page-table entries. (TLB 本质上就是一块高速缓存，类似 Cache,通过存储页表中的某些项，提高分页机制下的访问效率)



9.会计算分页方式下的内存有效访问时间？

答：实际上就是计算 TLB 命中和不命中两种情形下访问时间的均值

- **Associative Lookup = ε time unit**
- **Assume memory cycle time is 1 microsecond**
- **Hit ratio – percentage of times that a page number is found in the associative registers; ratio related to number of associative registers**
- **Hit ratio = α**
- **Effective Access Time (EAT)**

$$EAT = (1 + \varepsilon)\alpha + (2 + \varepsilon)(1 - \alpha)$$

$$= 2 + \varepsilon - \alpha$$

10.会计算页表大小？

答：实际上很简单，就是页表的项数（number of entry）乘上每一项的大小（32bit 地址空间下每一项的大小是 4byte）。

11.页表的具体实现有哪些手段，各有什么优缺点？

答：1. 分层页表 优点：减少了每一级页表所需的存储空间，缺点：增加了内存访问次数，需要更多的内存访问才能获得物理地址。

2. HASH 页表。（课本上没有提什么优缺点，如果非要说，可以说优点是提高了效率，缺点是实现复杂，还可能出现多个地址映射到 HASH 表中的同一行从而导致访问效率下降）。

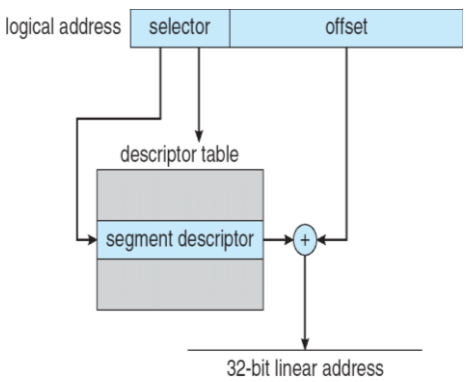
3. 反向页表。优点是系统只需要一个页表，减少了在每个进程总存储页表所需的内存空间。缺点是搜索页表的时间增加，不便于实现共享内存）（温馨提示：尽量多扯点）

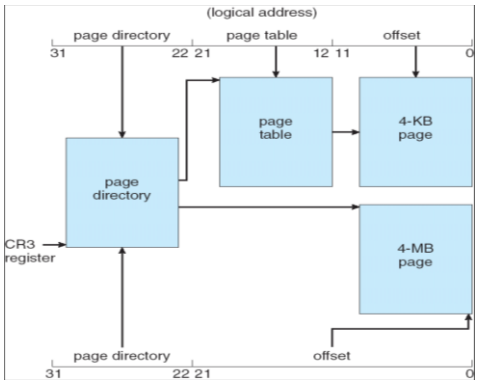
12.分段方式的机制是怎么样的和分页有什么区别？

答：参照 PPT 看下搞清楚概念即可。

13.会分析 X86 架构的段页式管理的地址转换过程？

答：





（温馨提示：会分析这两张图即可）

第九章 22 分

1.虚拟内存的基本原理是什么？为什么要提出虚拟内存

答：**Virtual memory** is a technique that allows the execution of processes that are not completely in memory. Virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory from the physical memory.

Why we need virtual memory? Only part of the program needs to be in memory for execution；

Logical address space can therefore be much larger than physical address space；Allows address spaces to be shared by several processes；Allows for more efficient process creation

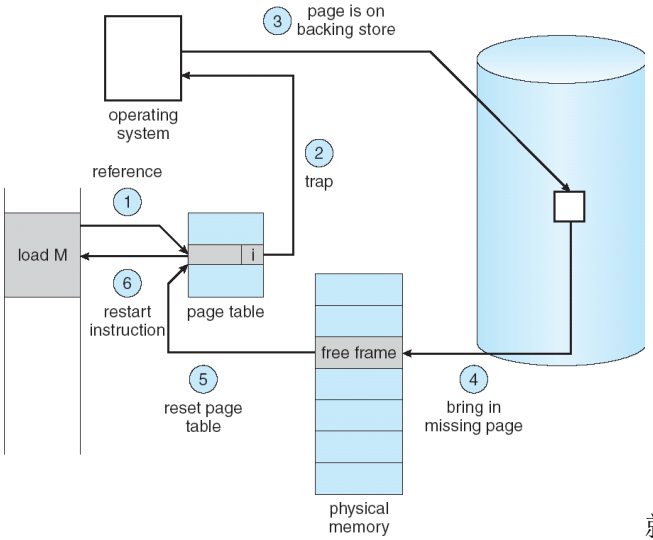
2.虚拟内存可以用哪些技术实现？

答：Virtual memory can be implemented via: Demand paging and Demand segmentation.

3.什么叫 Lazy swapper ？

答：**Lazy swapper** – never swaps a page into memory unless page will be needed.

4.会画 PAGEFAULT 的处理过程图



答：　　就是这张图了，记住就可以了。同时注意，这张图没有显示在系统没有空闲帧的时候的换入和换出的操作！）

5.会计算请求调页的有效访问时间

- **Page Fault Rate $0 \le p \le 1.0$**
  - **if $p = 0$ no page faults**
  - **if $p = 1$, every reference is a fault**

- **Effective Access Time (EAT)**

$$\begin{aligned} EAT = (1 - p) \times \text{memory access} \\ + p\ (\text{page fault overhead} \\ + \text{swap page out} \\ + \text{swap page in} \\ + \text{restart overhead} \\ ) \end{aligned}$$

答：

(温馨提示：计算内存访问时间的时候一定要搞清楚是几级页表！)

6.什么是页置换算法用在什么时候，什么是帧分配算法用在什么时候

答：**Frame-allocation algorithm**, if we have multiple processes in memory, we must decide how many frames to allocate to each process.

Page-replacement algorithms, if we have multiple processes in memory, we must decide how many frames to select the frames that are to be replaced.

7.FIFO 的思想，优缺点

答：FIFO，先入先出，将最先进入内存的页置换出去。优点是实现简单，缺点是实际中性能

较差，会导致 Belady's anomaly.

8.最优算法的思想，是否可实现，存在的意义

答：最有算法替换未来最长时间内不会被使用的页。最优算法难以实现，因为它需要知道未来页的引用情况。因此，最优算法的意义主要是与其他的替换算法进行比较研究。

9.LRU 的思想，优缺点，具体实现的机制

答：LRU,即最近最少使用。其核心思想是替换掉最后一次引用时间最靠前的页。优点是性能较 FIFO 明显提升。具体的实现方式可以使计数器或堆栈方式。

10.会采用三种算法计算 Page-Fault 次数

答：这道题我看了半天，也没明白是什么意思，灵机一动，我猜是不是说计算上面提到的三种页置换算法下的 Page-Fault 次数，如果是这样的话，就很简单了，耐心一点，一步步的把算法走一遍就好了。如果我的理解不对的话，这个真的不知道哪里冒出的三种计算 Page-Fault 的方法。（个人观点，谬误之处，恳请指正）

11.什么叫做系统颠簸，根本原因是什么？

答：**Thrashing** means a process is busy swapping pages in and out. It happens when a process does not have "enough" pages, the page-fault rate is very high, that means Σ size of locality > total memory size。

系统颠簸是指进程频繁的进行页的换入或换出。根本原因在于进程总的内存需求超过了物理内存的总数，进程频繁发生页请求错误。

12.局部模型用来干什么？

答：Locality model help us know how many frames a process need. The locality model states that, as a process executes, it moves from locality to locality. (帮助我们确定进程所需内存帧数，借助局部性原理)

13.工作集合模型怎样才能更接近局部模型？

答：工作集合模型是建立在局部性的假设下的，其关键在与选择 Δ，太大或太小都不好。

14.Page-Fault Frequency 和工作集合模型的区别与联系

答：PFF 更为直接，如果页请求错误概率过大，则说明进程需要更多的内存帧，否则说明进程拥有过多的内存帧。

15.内核空间为什么要采用单独的内存分配策略、分别有哪些？

答： 采用单独内存分配策略的原因：一，防止内部碎片，最大程度使用利用内核空间的内存。二，某些内核内存需要连续的物理内存空间。（Kernel requests memory for structures of varying sizes；Some kernel memory needs to be contiguous）分配策略包括： Buddy system 和 Slab allocation.

16.了解请求调页方式的其他问题，会计算不同循环顺序的 Page-Fault 次数。

- **Program structure**
  - `Int[128,128] data;`
  - **Each row is stored in one page**
  - **Program 1**

```
for (j = 0; j <128; j++)
    for (i = 0; i < 128; i++)
        data[i,j] = 0;
```

128 x 128 = 16,384 page faults

  - **Program 2**

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i,j] = 0;
```

128 page faults

答：

看明白这道题就可以了，可以联想 CACHE 的命中率问题，一样一样的，呵呵。

十、十一章 4 分 略