复习提纲

一、计算机系统概述

1.计算机基本构成：*处理器、内存、输入\输出设备、系统总线*。

2.处理器寄存器：a.**用户可见的寄存器**：可以用机器码访问存取，对所有程序都可用（应用/系统）（数据 DR，地址 AR，堆栈 SR，条件 CC）b.**控制和状态寄存器**：用以控制处理器的操作，且主要被系统程序使用。（程序计数器 PC，指令寄存器 IR）

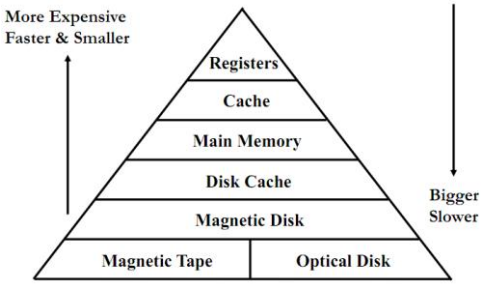
3.一条机器指令指定的 4 种操作：*处理器-寄存器；处理器-I/O；数据处理；控制*

4.**中断**：在一个进程进行过程中，当某事件发生时暂停流程，并在之后以类似的方式恢复流程。

5.多中断的处理方式：a.当正在处理一个中断时禁止再发生中断;b.定义中断优先级,允许高优先级的中断打断低优先级中断处理程序的运行.

6.内存层次的各个元素特征：价格递减，容量递增，存取时间递增，访问频率递减

7.**高速缓存**：在处理器和内存之间提供的一个容量小而速度快的存储器（采用与处理器寄存器相同构造的技术）



8.I/O 操作的三种技术 P23：

a.**可编程 I/O**:实施操作(不产生中断),处理器需不断查询 I/O 状态;

b.**中断驱动的 I/O**:以中断通知处理器,没有不必要的等待,但每个字节的读和写操作都通地过处理器消耗很多处理器时间;

c.**直接内存访问(DMA)**:直接传输内存中一块数据,处理器只在传输任务的开始和结束才参与

二、操作系统概述

1、操作系统提供的**服务**:

程序开发;程序运行;存取 I/O 设备;文件访问控制;系统访问;错误检测和响应;统计

2、操作系统***发展***:

串行处理→简单批处理→多道任务批处理→分时系统（人机交互）

3、操作系统***策略***:

进程:

内存管理:

信息防护和安全:

调度机制和资源管理:

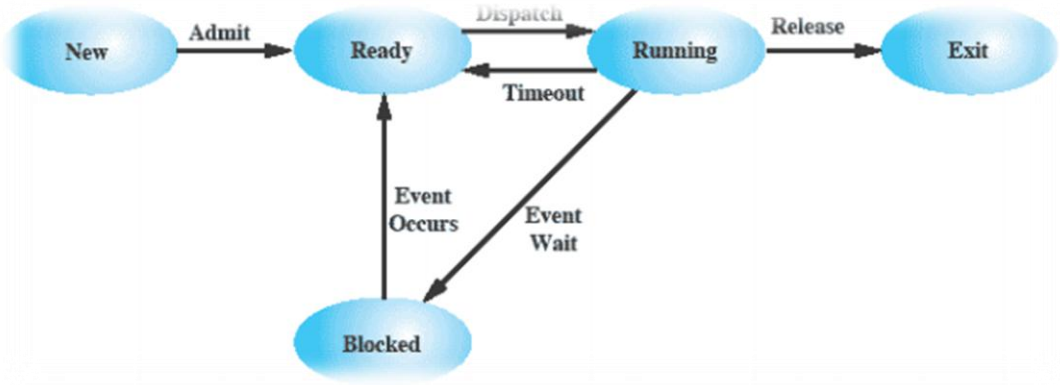
系统:

4、实地址和虚地址的区别:**虚地址**就是虚拟存储器中的地址，而虚拟存储器是允许程序从逻辑的角度访问存储器，而不考虑物理内存上可用的空间数量。**实地址**就是实际的内存中的地址。使用中，虚地址和实地址之间存在着动态的映射，当虚地址中内容在实际内存中不存在时，由存储管理硬件载入这块内容到内存。

5、单体内核和微内核的区别:P50 **单体内核**指操作系统应提供的功能由一个大内核提供（这些功能包括调度、文件系统、网络、设备驱动器、存储管理），典型情况下这个大内核作为一个进程实现所有元素共享相同的地址空间。**微内核**体系下则只给内核分配一些最基本的功能（*包括地址空间、进程间通信、基本的调度*），其他操作系统的服务都是由运行在用户态下与其他应用程序类似的进程提供。

三、进程描述和控制

1、五状态进程模型（P79，图 3.6）



—5 个状态（**运行**、**就绪**<进程做好准备，有机会就开始执行>、**阻塞**<进程在某些事件发生前不能执行>、**新建**<进程控制块已经创建但还没有加载到内存的新进程，没有加到可执行进程组中>、**退出**）

—6 个转换及转换条件：P79

空→新建：创建执行一个程序的新进程

新建→就绪：操作系统准备好接纳一个进程时，允许进入

就绪→运行：调度器选择一个就绪态的进程运行，分派

运行→退出：运行进程表示自己已经完成或者取消，释放

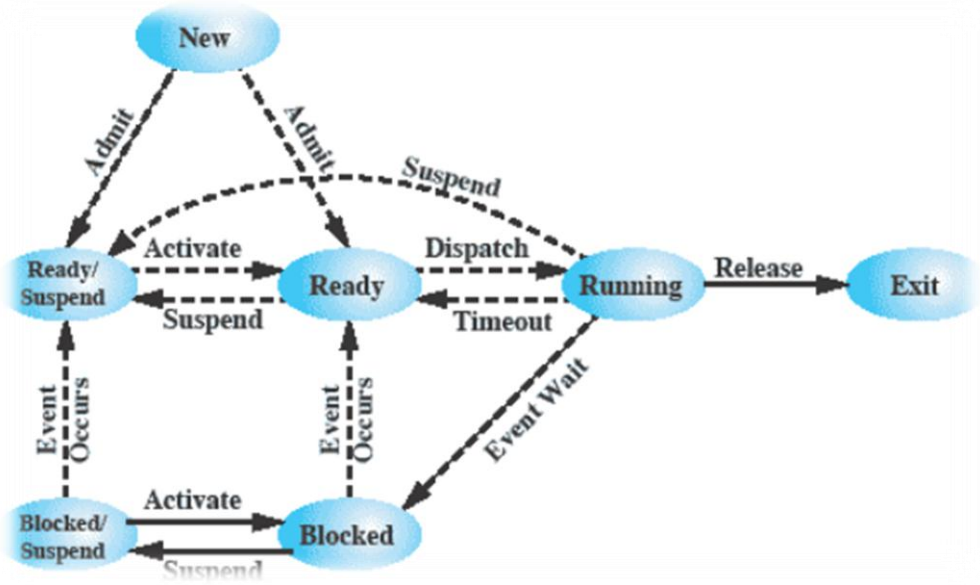
运行→就绪：达到了允许不中断执行的最大时间段，超时；优先级抢占；自愿释放控制

运行→阻塞：进程请求必须等待得某些事件

阻塞→就绪：所等待的事件发生

就绪→退出；阻塞→退出：某些系统中父进程可以终止子进程

2、有挂起态的进程状态转换图（P82，图 3.9b）



—7 个状态（add **阻塞/挂起态**<进程在外存中并等待事件>**就绪/挂起态**<进程在外存中但只要被载入内存即可执行>）

—13 个转换及转换条件：P83

3、哪些事件会导致创建一个进程？P77 表 3-1
新的批处理作业、交互登陆、操作系统因为提供一项服务而创建、由新进程派生。
4、进程控制块（PCB）中的典型元素：P87 表 3-5
进程标识信息：标识符：进程标识符 PID；创建进程的进程（父进程）标识符；用户标识符
处理器状态信息：用户可见寄存器；控制和状态寄存器 PSW（程序计数器、条件码、状态信息）；栈指针
进程控制信息：调度和状态信息；数据结构；进程间通信；进程特权；存储管理；资源所有权及使用情况
5、什么是交换，其目的是什么？
操作系统把内存中某个进程的一部分或全部移到磁盘中；当内存中没有处于就绪态的进程时，把被阻塞的进程换出到磁盘中的挂起队列。可以释放出内存并使处理器可以处理其他更多的进程。
6、操作系统创建一个新进程的步骤？（P91）
1.给新进程分配一个唯一的进程标识符 2.给进程分配空间 3.初始化进程控制块
4.设置正确的连接 5.创建或扩充其他数据结构
7、进程切换的步骤？（P93）
1.保存处理器上下文，包括 PC 和各寄存器 2.更新进程 PCB，由 Running 修改为其他状态
3.将此 PCB 移至相应队列 4.选择另一个需要运行的进程 5.更新被选中的进程 PCB
6.更新内存管理相关的数据结构 7.恢复被选中进程相关的 context
8、P103 习题 3.2：计算机有 A 输入输出设备 B 处理器，最多容纳 C 进程（A<B<C）
a)任一时刻，**的最大进程数目：处于就绪态（C-B）、运行态（B）、阻塞态（C）、就绪挂起态（无限制）、阻塞挂起态（无限制）
b)任一时刻最小进程数目：全部为 0

四、线程和微内核

1、名词解释：
—**内核级线程（KLT）**：各进程和线程的上下文信息由内核管理，应用程序部分没有进行线程管理的代买，调度基于 thread 完成。
—**用户级线程（ULT）**：线程所有的管理由应用程序完成，内核意识不到线程的存在。
内核可以同时在多个处理器上调度一个进程的多线程；如果一个处理器上的线程阻塞，内核可以调度同进程的其他线程；内核程序本身也可以是多线程。但是，同一个进程的多个线程之间的切换，需要切换为内核模式。
2、列举使用***线程***相对使用***进程***的好处
1.开销小、时间花费少，如表现在创建新线程、线程切换、终止线程上。
2.资源共享性好，通信便利高效，进程间通信需要内核介入，而一个进程内的线程共享内存和文件，直接可以通信。
附缺点：1.共享资源需要耗费一定的锁资源，同步相对复杂；2.一个线程崩溃可能导致整个进程崩溃
3、对比微内核操作系统和单体内核操作系统，7 个优点和可能存在的性能缺点
优点：1.为进程发出的请求提供一致接口，不需要区分内核级/用户机，都提供消息传递；
2.可扩展性，允许增加新的服务以及同一功能区提供多个服务；
3.灵活性，增加新功能&删减功能以更小有效的实现；
4.可移植性；5.可靠性；6.支持分布式系统；7.支持面向对象的操作系统
缺点：性能问题：耗时更大--解决办法：把一些关键的服务程序和驱动程序重新放回内核；

五、并发：互斥和同步

1、名词解释：

—**临界区**：一段代码，在这段代码中进程将访问共享资源，当另一个进程已经在这段代码中运行时，这个进程就不能在这段代码中执行。

—**死锁**：两个或两个以上的进程因其中的每个进程都在等待其他进程做完某些事情而不能继续执行，这样的情形叫做死锁。

—**竞争条件**：多个线程或进程在读写一个共享数据时，结果依赖于它们执行的相对时间。

—**信号量**：用于进程间传递信号的一个整数值,只有三种(原子)操作：初始化、递减、增加

2、互斥的实现方法

1.**硬件支持**：中断禁用(单处理器仅支持交叉、进程持续运行直到它发出系统调用或被中断、禁止中断可保证互斥、在多处理器架构中是无效的);专用机器指令(compare&exchange/exchange)

2.**信号量**：P150 第 1、2 段

3.**管程**：管程中的数据变量每次只能被一个进程访问到。

4.**信息传递**：消息函数（令牌）：希望进入临界区的进程首先试图接受一条消息，如果信箱为空则被阻塞；一旦进程获得消息则执行其临界区并把该消息放回信箱。

```
void P1,P2,...
 {
 while(ture){
  /*dealing codes*/
 entercritical(Ra);
  /*critical section*/
 entercritical(Ra);
  /*remainder*/
 }
```

```
while (true) {
   /* disable interrupts */;
   /* critical section */;
   /* enable interrupts */;
   /* remainder */;
}
```

```
/* program mutualexclusion */
const int n = /* number of processes */;
int bolt;
void P(int i)
{
   while (true) {
     while (compare_and_swap(bolt, 0, 1) == 1)
          /* do nothing */;
      /* critical section */;
      bolt = 0;
      /* remainder */;
   }
}
void main()
{
   bolt = 0;
   parbegin (P(1), P(2), ... ,P(n));

}
```

```
   /* program mutualexclusion */
int const n = /* number of processes**/;
int bolt;
void P(int i)
{
   int keyi = 1;
   while (true) {
      do exchange (keyi, bolt)
      while (keyi != 0);
      /* critical section */;
      bolt = 0;
      /* remainder */;
   }
}
void main()
{
   bolt = 0;
   parbegin (P(1), P(2), ..., P(n));
}
```

```
/* program mutualexclusion */
const int n = /* number of processes  */;
semaphore s = 1;
void P(int i)
{
     while (true) {
          semWait(s);
          /* critical section    */;
          semSignal(s);
          /* remainder    */;
     }
}
void main()
{
     parbegin (P(1), P(2), . . ., P(n));
}
```

```
/* program mutualexclusion */
const int n = /* number of processes  */;
void P(int i)
{
    message msg;
    while (true) {
     receive (box, msg);
     /* critical section    */;
     send (box, msg);
     /* remainder    */;
    }
}
void main()
{
    create mailbox (box);
    send (box, null);
    parbegin (P(1), P(2), . . ., P(n));
}
```

3、习题 5.4 使用通用信号量完成程序，使得进程总是以 A,B,A,A 的顺序结束：

progame As_and_Bs

var {/*信号量声明*/semaphore sa=1,sb=0, b_end_flag=0; }

procedure A();

begin

{/*仅填入 P、V 语句*/

```
    semWait(sa);
    if(b_end_flag==0)semSignal(sb);
    else semSignal(sa);}
end
begin
    {/*仅填入 P、V 语句*/
    semWait(sb);
    b_end_flag=1;
    semSignal(sa);}
end
begin(*main program*)
    parbegin
      A();A();A();B();
    parend
end
eg5.21 program    producer consumer;
var       n: integer;
          s: (*binary*) semaphore (:= 1);
          delay: (*binary*) semaphore (:= 0);
procedure producer;
begin
  repeat
    produce;
    semWaitB(s);
    append;
    n := n + 1;
    if n=0 then semSignalB(delay);
    semSignalB(s)
  forever
end;
procedure consumer;
begin
  repeat
    semWaitB(s);
    take;
    n := n – 1;
    if n = -1 then
       begin
       semSignalB(s);
       semWaitB(delay);
       semWaitB(s)
       end;
    consume;
    semSignalB(s)
```

这部分代码的书写
具体可以参考一下后面示例5.21

```
    forever
end;
begin (*main program*)
    n := 0;
    parbegin
        producer; consumer
    parend
end.
```

六、并发：死锁和饥饿

1、死锁发生的 4 个条件(**互斥**:一次只有一个进程可以使用一个资源;**占有且等待**:当一个进程等待其他进程时，继续占有已经非配的资源;**不可抢占**:不能强行抢占进程已占有的资源;**循环等待**:存在一个封闭的进程链，使得每个进程至少占有此链中下一个进程所需资源)

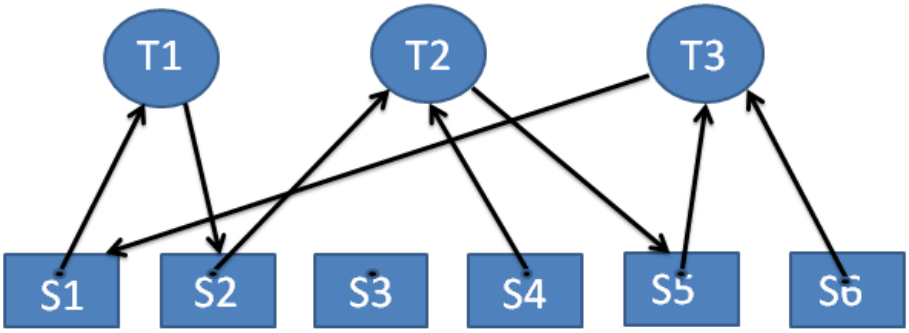
—如何防止*占有且等待*:要求进程一次请求全部他所需要的资源

—如何防止*不可抢占*: 进程必须释放资源，下次再请求；允许 OS 强制要求进程释放资源。只有在资源状态可以很容易恢复的情况下，这种方法才实用。

—如何防止*循环等待*: 将资源类型定义为线性顺序，如果进程已经分配到了 R 类型的资源，那么它接下来请求的资源只能是排在 R 类型之后的资源类型。

2、（P205）习题 6.5 画出资源分配图 P182

eg　T1 拥有 s1,请求 s2;T2 拥有 s2、s4 请求 s5;T3 拥有 s5、s6 请求 s1



3、习题 6.6 死锁检测 P189

Available=(2 1 0 0), $\mathrm{Re}\,quest = \begin{bmatrix} 2&0&0&1 \\ 1&0&1&0 \\ 2&1&0&0 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \end{matrix}$, $Allocation = \begin{bmatrix} 0&0&1&0 \\ 2&0&0&1 \\ 0&1&2&0 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \end{matrix}$

(1)Begin:W=(2 1 0 0)　(2)Mark P3:W=(2 2 2 0)　(3)Makr P2:W=(4 2 2 1)　(4)Mark P1:W=(4 2 3 1)

4、习题 6.11 银行家算法 P186

$Claim = \begin{bmatrix} 4&4&2&1 \\ 4&3&1&1 \\ 13&5&2&7 \\ 6&1&1&1 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$, $Allocation = \begin{bmatrix} 4&0&0&1 \\ 1&2&1&0 \\ 1&1&0&2 \\ 3&1&1&0 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$,resource=[16 5 2 8]

Ⅰ.(1)begin: resource=[16 5 2 8],available=[7 1 0 5]

$Claim = \begin{bmatrix} 4&4&2&1 \\ 4&3&1&1 \\ 13&5&2&7 \\ 6&1&1&1 \end{bmatrix}$, $Allocation = \begin{bmatrix} 4&0&0&1 \\ 1&2&1&0 \\ 1&1&0&2 \\ 3&1&1&0 \end{bmatrix}$, $C-A = \begin{bmatrix} 0&4&2&0 \\ 3&1&0&1 \\ 12&4&2&5 \\ 3&0&0&1 \end{bmatrix}$

(2)runP2: resource=[16 5 2 8],available=[8 3 1 5]

$$Claim=\begin{bmatrix}4&4&2&1\\0&0&0&0\\13&5&2&7\\6&1&1&1\end{bmatrix},\ Allocation=\begin{bmatrix}4&0&0&1\\0&0&0&0\\1&1&0&2\\3&1&1&0\end{bmatrix},\ C-A=\begin{bmatrix}0&4&2&0\\0&0&0&0\\12&4&2&5\\3&0&0&1\end{bmatrix}$$

(3)runP4: resource=[16 5 2 8],available=[11 4 2 5]

$$Claim=\begin{bmatrix}4&4&2&1\\0&0&0&0\\13&5&2&7\\0&0&0&0\end{bmatrix},\ Allocation=\begin{bmatrix}4&0&0&1\\0&0&0&0\\1&1&0&2\\0&0&0&0\end{bmatrix},\ C-A=\begin{bmatrix}0&4&2&0\\0&0&0&0\\12&4&2&5\\0&0&0&0\end{bmatrix}$$

(4)runP1: resource=[16 5 2 8],available=[15 4 2 6]

$$Claim=\begin{bmatrix}0&0&0&0\\0&0&0&0\\13&5&2&7\\0&0&0&0\end{bmatrix},\ Allocation=\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&1&0&2\\0&0&0&0\end{bmatrix},\ C-A=\begin{bmatrix}0&0&0&0\\0&0&0&0\\12&4&2&5\\0&0&0&0\end{bmatrix}$$

(5)runP3=>safe

Ⅱ.P1 request 1 unit of R2

resource=[16 5 2 8],available=[7 0 0 5]

$$Claim=\begin{bmatrix}4&4&2&1\\4&3&1&1\\13&5&2&7\\6&1&1&1\end{bmatrix},\ Allocation=\begin{bmatrix}4&1&0&1\\1&2&1&0\\1&1&0&2\\3&1&1&0\end{bmatrix},\ C-A=\begin{bmatrix}0&3&2&0\\3&1&0&1\\12&4&2&5\\3&0&0&1\end{bmatrix}$$

runP4:resource=[16 5 2 8],available=[10 1 1 5]

$$Claim=\begin{bmatrix}4&4&2&1\\4&3&1&1\\13&5&2&7\\0&0&0&0\end{bmatrix},\ Allocation=\begin{bmatrix}4&1&0&1\\1&2&1&0\\1&1&0&2\\0&0&0&0\end{bmatrix},\ C-A=\begin{bmatrix}0&3&2&0\\3&1&0&1\\12&4&2&5\\0&0&0&0\end{bmatrix}$$

=>runP2=>runP1=>runP3=>safe

Ⅲ.P3 request 6 units of R1

resource=[16 5 2 8],available=[1 1 0 5]

$$Claim=\begin{bmatrix}4&4&2&1\\4&3&1&1\\13&5&2&7\\6&1&1&1\end{bmatrix},\ Allocation=\begin{bmatrix}4&0&0&1\\1&2&1&0\\7&1&0&2\\3&1&1&0\end{bmatrix},\ C-A=\begin{bmatrix}0&4&2&0\\3&1&0&1\\6&4&2&5\\3&0&0&1\end{bmatrix}$$

=>unsafe

Ⅳ.P2 request 2 unit of R4

resource=[16 5 2 8],available=[7 1 0 3]

$$Claim=\begin{bmatrix}4&4&2&1\\4&3&1&1\\13&5&2&7\\6&1&1&1\end{bmatrix},\ Allocation=\begin{bmatrix}4&0&0&1\\1&2&1&2\\1&1&0&2\\3&1&1&0\end{bmatrix},\ C-A=\begin{bmatrix}0&4&2&0\\3&1&0&-1?\\12&4&2&5\\3&0&0&1\end{bmatrix}$$

=>runP2=>...=>safe

七、内存管理

1、**重定位**，为什么需要重定位进程的能力

编程人员不知道程序运行时放在内存的哪个位置。程序可能交换到磁盘，再调入到内存中不同的位置，因此需要重定位。

2、**内部碎片**：由于被装入的数据小于分区的大小，从而导致分区内部空间浪费的现象。

外部碎片：动态分区随时间推移导致内存中出现越来越多的空洞，内存利用率随之下降的现象。

3、习题 7.2 **固定分区** P213 最佳适应（每个分区一个调度队列）最佳可得（只提供一个调度队列）

part0=20K,part1=30K,part3=50K

进程:所需内存,时间 p1:18k,10ms; p2:29k,5ms; p3:10k,20ms; p4:33k,15ms; p5:14k,15ms; p6:49k,10ms

a.最佳适应分配:分配到最小的分区

part0:p1:10ms;p3:20ms;p5:15ms　part1:p2:5ms part3:-- part4:p4:15ms;p6:10ms　共 45ms

b.最佳可得分配:分配到最小的空闲分区

part0:p1:10ms　part1:p2:5ms;p5:15ms　part2:p3:20ms　part3:p4:15ms;p6:10ms 共 25ms

4、习题 7.6 **动态分区** P216，最佳适配、首次适配、下次适配、~~最差适配~~

(MB)20+ 20- 40+ 60- 20+ 10- 60+ 40- 20+ 30- 40+ 40-(+占有-空闲)　接下来:40\20\10

最佳适配：20+ 20-(20) 40+ 60- 20+ 10-(10) 60+ 40-(40) 20+ 30- 40+ 40-　(230 20 160)

首次适配：20+ 20-(20) 40+ 60-(40+10) 20+ 10- 60+ 40- 20+ 30- 40+ 40-　(80 20 120)

下次适配：20+ 20- 40+ 60-(40+20) 20+ 10-(10) 60+ 40- 20+ 30- 40+ 40-　(80 120 160)

最差适配：20+ 20- 40+ 60-(40) 20+ 10- 60+ 40-(20) 20+ 30- 40+ 40-(10)　(80 230 360)

5、习题 7.7 **伙伴系统** P217

A 申请 25k B 申请 500k C 申请 60k D 申请 100k E 申请 30k&释放 A F 申请 20k

	1024					
请求A=25K	A=32	64K	128K	256K		512K
请求B=500K	A=32	64K	128K	256K		B=512K
请求C=60K	A=32	C=64K	128K	256K		B=512K
请求D=100K	A=32	C=64K	D=128K	256K		B=512K
请求E=30K	A=E=	C=64K	D=128K	256K		B=512K
释放A	32	E=	C=64K	D=128K	256K	B=512K
申请F=20K	F=	E=	C=64K	D=128K	256K	B=512K

6、习题 7.12 **页表** P236　逻辑地址（页号，偏移量）　物理地址（页框号，偏移量）

32 位机器使用三级页表，用 6 位标识页号(此处应为页框号)。逻辑地址分为 8 位、6 位、6 位、12 位四部分。

该系统页的大小:2^12=4kb　能够分配给一个进程的最多的页面个数:2^(32-12)=1MB

最大逻辑地址空间:2^32=4GB 物理内存的大小:2^(6+12)=256KB

7、习题 7.14 **分段** P222

分段系统：起始地址:长度 660:246;1752:442;222:198;996:604

a.0,198:660+198=858　b.2,156:222+156=378　c.1,530:530>442->error

d.3,444 996+444=1440　e.0,222:660+222=882

八、虚拟内存

1、**驻留集**：进程执行中任何时候都在在内存中的部分。

2、**抖动**：处理器大部分时间都用于交换内存块而不是在执行指令。

3、**转换检测缓冲区（TLB）**：为页表项 PTE 使用的特殊高速缓存，包含最近使用过的页表项

4、习题 8.2 **多级页表**设计 P236

32 位地址、1KB 页大小的分页虚拟内存系统。每个页表项需要 32 位，限制页表大小为一个页

a.1KB=2^10 页表项,2^10/4=2^8 页表, 4GB(2^32)虚拟地址由 2^22 页组成=>分级　一共需要 3 级；

b.两级页表包含 2^8 个页表项;一级页表包含 2^6 个页表项(8+8+6=22).

c.在第一级使用的页较小与在最底下一级使用的页较小相比，哪种策略使用最小个数的叶？

Less space is consumed if the top level has 2^6 entries. In that case, the second level has 2^6 pages with 2^8 entries each, and the bottom level has 2^{14} pages with 2^8 entries each, for a total of $1 + 2^6$

+ 2^{14} pages = 16,449 pages.

If the middle level has 2^6 entries, then the number of pages is 1 + 2^8 + 2^{14} pages = 16,641 pages.

If the bottom level has 2^6 entries, then the number of tables is 1 + 2^8 + 2^{16} pages = 65,973 pages.

5、习题 8.4 **置换算法** P247:FIFO,LRU,时钟算法,~~OPT~~

虚拟页号	页框	加载时间	(上一次)访问时间	访问 R 位	修改 M 位
2	0	60	161	0	1
1	1	130	160	0	0
0	2	26	162	1	0
3	3	20	163	1	1

虚拟页 4 发生缺页中断时:

a.**FIFO**:PFN3 since loaded longest ago at time 20;

b.LRU:PFN1 since referenced longest ago at time 160;

c.**Clock**: 1)选择遇到的第一个 r=0m=0 用于置换；2)重新扫描 r=0m=1 (跳过的设置 r=0)；3)回到最初位置重新(设置所有 r=0)扫描 1)&2)

Clear R in PFN3(oldest loaded), clear R in PFN2(next oldest loaded), victim PFN is 0 since R=0;

d.**OPT**： Replace the page in PFN 3 since VPN 3 (in PFN 3) is used furthest in the future;

~~e.工作集~~ 4,0,0,0,2,4,2,1,0,3,2

6、习题 8.6 FIFO，LRU，OPT 计算缺页次数

460 字节程序的虚拟地址序列：10,11,104,170,73,309,185,245,246,434,458,364

a.假设页面大小为 100 字节，**引用串**：0,0,1,1,0,3,1,2,2,4,4,3

b.1.LRU:5 次缺页

	0	0	1	1	0	3	1	2	2	4	4	3
0	0	0	0	0	0	0	1	1	1	4	4	4
1			1	1	1	3	3	2	2	2	2	3

2.FIFO:4 次缺页

	0	0	1	1	0	3	1	2	2	4	4	3
0	0	0	0	0	0	3	3	3	3	4	4	4
1			1	1	1	1	1	2	2	2	2	3

3.OPT: 3 次缺页

	0	0	1	1	0	3	1	2	2	4	4	3
0	0	0	0	0	0	3	3	3	3	3	3	3
1			1	1	1	1	1	2	2	4	4	4

7、习题 8.10 64 位系统，页面大小 4kb

a.虚拟地址空间: 2^{64};b.一级页表有 2^{52} 表项;c.b 存在的问题:表项太多维护难->多级页表

Since each page table entry is 4 bytes and each page contains 4 Kbytes, then a one-page page table would point to 1024=2^{10} pages, addressing a total of $2^{10} \times 2^{12} = 2^{22}$ bytes. The address space however is 2^{64} bytes. Adding a second layer of page tables, the top page table would point to 2^{10} page tables, addressing a total of 2^{32} bytes. Continuing this process, we can see that 5 levels do not address the full 64 bit address space, so a 6th level is required. But only 2 bits of the 6th level are required, not the entire 10 bits. So instead of requiring your virtual addresses be 72 bits long, you could mask out and ignore all but the 2 lowest order bits of the 6th level. This would give you a 64 bit ddress. Your top level page table then would have only 4 entries. Yet another option is to revise the criteria that the top level page table fit into a single physical page and instead make it fit into 4 pages. This would save a physical page, which is not much.

8、习题 8.11 **使用 TLB 的分页系统**的计算时间

二级页表，TLB 存储最近访问 16 个页表表项。访问内存需要 80ns，TLB 检查需要 20ns，页面交换需要 5000ns。假如 20%页面置换被更改，TLB 命中率 95%，缺页率 10%。

1）TLB 检查(20ns)并命中(95%)，直接访问内存中的页面(80ns)：(20+80)*95%=95ns

2）TLB 检查(20ns)未命中(5%)，直接访问内存中的一级页表(80ns)，通过一级页表访问二级页表(80ns)，页面保存在内存中(90%)，直接访问内存中的页面(80ns)：

(20+(80+80+80)*90%)*5%=11.8ns

3）TLB 检查(20ns)未命中(5%)，直接访问内存中的一级页表(80ns)，通过一级页表访问二级页表(80ns)，发生缺页中断(10%)，被替换的页面未更改内容(80%)，执行一次页面交换(5000ns)，最后直接访问内存中的页面(80ns)：(20+(80+80+5000*80%+80)*10%)*5%=22.2ns

4）TLB 检查(20ns)未命中(5%)，直接访问内存中的一级页表(80ns)，通过一级页表访问二级页表(80ns)，发生缺页中断(10%)，被替换的页面被更改内容(80%)，执行两次页面交换(5000ns*2)，最后直接访问内存中的页面(80ns)：

(20+(80+80+5000*2*20%+80)*10%)*5%=11.4ns

综上所述，访问一个数据需要时间：140.4ns。

九、单处理器调度

1、**FCFS 调度、RR 轮转调度、SPN 最短进程优先调度、SRT 最短剩余时间调度、HRRN 最高响应比优先调度** 2、**反馈调度**→习题 9.1&9.2

3、习题 9.1，9.2 给定进程到达时间，处理时间表格，画出调度图 详见 P279

9.1 进程：到达时间、处理时间 A:0\3 B:1\5 C:3\2 D:9\5 E:12\5

		0 5 10 15 20	Ta	Ts	Tf	Tr	Tr/Ts	
FCFS	A		0	3	3	3	1	
	B		1	5	8	7	1.4	
	C		3	2	10	7	3.5	
	D		9	5	15	6	1.2	~Tr=6.2
	E		12	5	20	8	1.6	~Tr/Ts=1.74
RR, q=1	A		0	3	6	6	2	
	B		1	5	11	10	2	
	C		3	2	8	5	2.5	
	D		9	5	18	9	1.8	~Tr=7.6
	E		12	5	20	8	1.6	~Tr/Ts=1.78
RR, q=4	A		0	3	3	3	1	
	B		1	5	10	9	1.8	
	C		3	2	9	6	3	
	D		9	5	19	10	2	~Tr=7.2
	E		12	5	20	8	1.6	~Tr/Ts=1.88
SPN	A		0	3	3	3	1	
	B		1	5	10	9	1.8	
	C		3	2	5	2	1	
	D		9	5	15	6	1.2	~Tr=5.6
	E		12	5	20	8	1.6	~Tr/Ts=1.32
SPT	A		0	3	3	3	1	
	B		1	5	10	9	1.8	
	C		3	2	5	2	1	
	D		9	5	15	6	1.2	~Tr=5.6
	E		12	5	20	8	1.6	~Tr/Ts=1.32
HRRN	A		0	3	3	3	1	
	B		1	5	8	7	1.4	
	C		3	2	10	7	3.5	
	D		9	5	15	6	1.2	~Tr=6.2
	E		12	5	20	8	1.6	~Tr/Ts=1.74
Fb, q=1	A	0 1 2	0	3	7	7	2.33	
	B	0 1 2 3 4	1	5	18	17	3.4	
	C	0 1	3	2	6	3	1.5	
	D	0 1 2 3 4	9	5	19	10	2	~Tr=9
	E	0 1 2 3 4	12	5	20	8	1.6	~Tr/Ts=2.17
Fb, q=2^i	A	0 1	0	3	4	4	1.33	
	B	0 1 2	1	5	10	9	1.8	
	C	0 1	3	2	8	5	2.5	
	D	0 1 2	9	5	18	9	1.8	~Tr=7
	E	0 1 2	12	5	20	8	1.6	~Tr/Ts=1.81

反馈调度的策略有几种理解，这里应该有争议

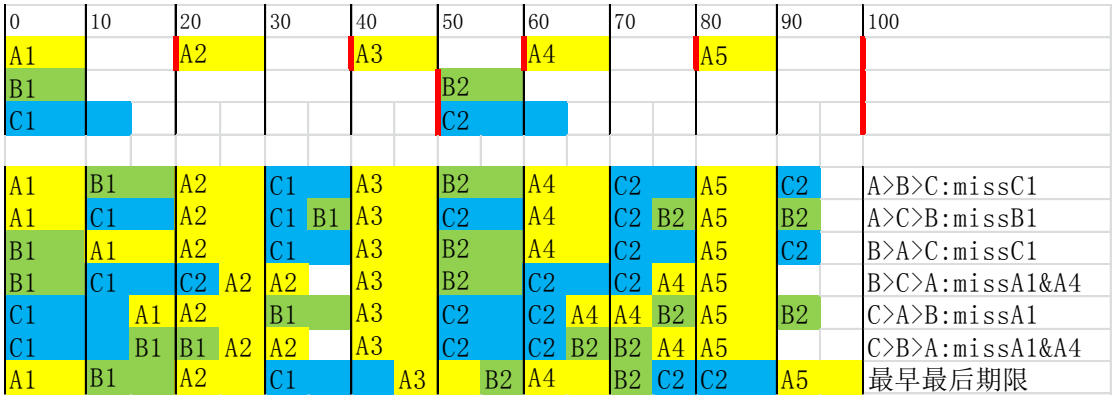
十、实时调度 P318

1、习题 10.1 具有**完成 deadline** 的周期任务，画出：

—**固定优先级**调度图　　—使用**最早完成 deadline** 调度图

周期任务：进程：到达时间、执行时间、完成最后期限：

A(1):0|10|20;A(2):20|10|40…B(1):0|10|50;B(2):50|10|100…C(1):0|15|50;C(2):50|15|100…

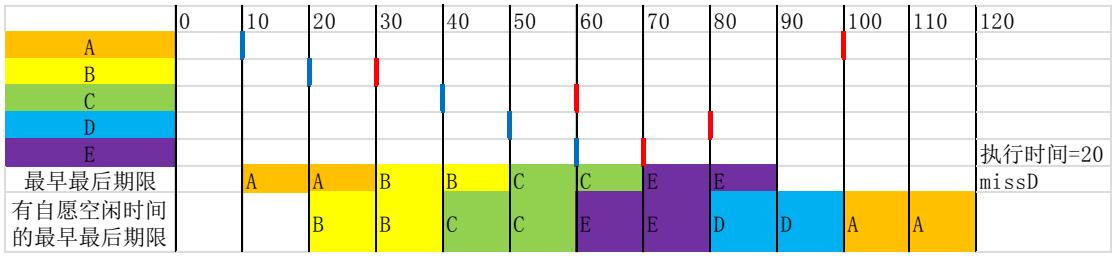


2、习题 10.2 具有**启动 deadline** 的非周期任务，画出：

—最早启动 deadline 调度图 —有**自愿空闲时间的最早 deadline** 调度图

非周期任务：进程：到达时间、执行时间、启动最后期限：

A:10|20|100 B:20|20|30 C:40|20|60 D:50|20|80 E



总是调度最早期限最早的合格任务，并让该任务运行直到完成。

十一、I/O 管理

1、**逻辑 I/O**：代表用户进程管理的一般的 I/O 功能，用户进程根据设备标识符以及如打开、关闭、读、写之类的简单命令与设备打交道；**设备 I/O**：请求的操作被转换成适当的 I/O 指令序列、通道命令和控制器指令。（可以使用缓冲技术提高其使用率）

2、**面向块的设备**将信息保存在固定大小的块中，传输过程中一次传送一块，如磁盘、USB keys；**面向流的设备**以字节流的方式输入输出数据没有块结构，如终端、打印机、通信端口、鼠标以及大多数的非辅存设备。

3、习题 11.1 一个程序访问一个 IO 设备，比较无缓冲和使用缓冲的 IO，说明**使用缓冲区最多可以减少 2 倍的运行时间**：如果计算的时间正好等于它的 I/O 时间（它是最佳环境），操作者和外围设备同时运行。如果单独运行，只要花费他们的一半时间，设 C 是整个程序的计算时间，T 为所要求总的 I/O 时间，因而寄存器最好的运行时间是 max(C,T)，不需要寄存器的运行时间是 C+T，显然（(C+T)/2）≤max(C,T)≤(C+T).

十三、嵌入式操作系统

1、**嵌入式系统**：为了完成某个特定功能而设计的，或许附加其他机制或其它部分，计算机硬件和软件的结合体。

2、嵌入式操作系统的关键*特点*：实时操作；响应操作；可配置性；I/O 设备的灵活性；改进的保护机制；直接使用中断。