习题答案

Chapter 1-12

1.计算机系统概述

- **复习题1.1**

  处理器，它控制计算机的操作并执行数据处理功能；

  主存储器，它存储数据和指令；

  I/O模块，它在计算机和外部环境之间传输数据；

  以及系统总线，它提供处理器、主存储器和I/O模块之间的通信。

1.计算机系统概述

- **复习题1.4**

  中断是一种机制，通过该机制其他模块（I/O，内存）可以中断处理器的正常顺序。

1.计算机系统概述

- **复习题1.7**

  高速缓存是一种比主存储器更小更快的内存，位于处理器和主存储器之间。高速缓存充当最近使用的内存位置的缓冲区。

1.计算机系统概述

- **习题1.1**

内存（十六进制内容）：300:3005; 301:5940; 302:7006

步骤 1: 3005 → IR;

步骤 2: 3 → AC

步骤 3: 5940 → IR;

步骤 4: 3 + 2 = 5 → AC

步骤 5: 7006 → IR;

步骤 6: AC → 设备6

1.计算机系统概述

- **习题1.8**

我们忽略数据读/写操作，假设处理器只取指令。那么处理器每微秒需要访问一次主存。DMA模块以每秒1200个字符的速率传输字符，即每833微秒一个。因此，DMA每833个周期“窃取”一次。这使处理器的速度大约降低了0.12%。

1.计算机系统概述

- **习题1.13**

这里有三种情况需要考虑：

被引用词的位置	概率	总访问时间（纳秒）
在缓存中	0.9	20
不在缓存中，但在主存中	(0.1)(0.6)=0.06	60 + 20 = 80
不在缓存或主存中	(0.1)(0.4)=0.04	12ms + 60 + 20 = 12,000,080

因此，平均访问时间为： 平均 =(0.9)(20)+(0.06)(80)+(0.04)(12000080)= 480026 纳秒

2.操作系统概述

- 复习题2.1

便利性：操作系统使计算机使用起来更方便。

效率：操作系统允许计算机系统资源以高效的方式使用。

可演化性：操作系统应以某种方式构建，以便允许有效开发、测试和引入新系统功能，而不会干扰服务。

2.操作系统概述

- 复习题2.3

多道程序设计是一种操作模式，提供单个处理器对两个或多个计算机程序的交错执行。

2.操作系统概述

- 复习题2.5

执行上下文或进程状态是操作系统能够监督和控制进程的内部数据。这些内部信息与进程分离，因为操作系统拥有进程无法访问的信息。上下文包括操作系统管理进程和处理器正确执行进程所需的所有信息。上下文包括各种处理器寄存器的内容，如程序计数器和数据寄存器。它还包括操作系统使用的信息，如进程的优先级以及进程是否在等待特定I/O事件的完成。

2.操作系统概述

- 习题2.3

在分时系统中，关注点是周转时间。时间片轮转被优先考虑，因为它在短时间内给予所有进程对处理器的访问。在批处理系统中，关注点是吞吐量，越少的上下文切换，进程可用的处理时间就越多。因此，倾向于减少上下文切换的策略。

2.操作系统概述

- 习题2.4

系统调用是应用程序用来调用操作系统提供的功能。通常，系统调用会导致切换到在内核模式下运行的系统程序。

3.进程描述和控制

- 复习题3.3

运行状态：当前正在执行的进程。

就绪状态：当有机会执行时准备执行的进程。

阻塞状态：由于某些事件（如I/O操作的完成）而无法执行的进程。

新建状态：刚刚创建但尚未被操作系统接纳到可执行进程池中的进程。

退出状态：已被操作系统从可执行进程池中释放的进程，可能是因为已停止或因某种原因中止。

3.进程描述和控制

- 复习题3.9

进程识别、处理器状态信息和进程控制信息。

3.进程描述和控制

- 复习题3.14

模式切换可能发生在不改变当前运行状态进程的情况下。

进程切换涉及将当前执行的进程从运行状态移出，转而执行另一个进程。进程切换涉及保存更多的状态信息。

3.进程描述和控制

- 习题3.12

0
<child pid>

或

<child pid>
0

5.并发：互斥与同步

- **复习题5.5**

竞争进程需要同时访问相同的资源，例如磁盘、文件或打印机。合作进程要么共享对一个公共对象（例如内存缓冲区）的访问，要么能够相互通信，并在某些应用或活动的执行中合作。

5.并发：互斥与同步

- **复习题5.7**

1.必须执行互斥：在所有对同一资源或共享对象有临界区的进程中，一次只允许一个进程进入其临界区。

2. 在非临界区停止的进程必须在不干扰其他进程的情况下进行。

3. 需要访问临界区的进程不应无限期延迟：没有死锁或饥饿。

4. 当没有进程在临界区时，任何请求进入其临界区的进程都必须立即被允许进入。

5. 不做关于相对进程速度或处理器数量的假设。

6. 进程只在其临界区内停留有限的时间。

5.并发：互斥与同步

- **复习题5.8**

1. 信号量可以初始化为非负值。

2. wait操作递减信号量值。如果值变为负数，则执行wait的进程被阻塞。

3. signal操作递增信号量值。如果值不是正数，则被wait操作阻塞的进程被解锁。

5.并发：互斥与同步

- **习题5.6**

a. 对于“x is 10”，找到产生所需行为的交错很容易，因为它只需要在源语言语句级别进行交错。这里的基本事实是，x的值的测试与另一个进程对x的增量交错进行。因此，执行测试时x不等于10，但读取内存中的x值进行打印时等于10。

	M(X)
P1:x = x - 1;	**9**
P1:x = x + 1;	**10**
P2:x = x- 1;	**9**
P1:if(x != 10)	**9**
P2:x = x + 1;	**10**
P1:printf("x is %d",x);	**10**

"x is 10" is printed.

5.并发：互斥与同步

- **习题5.6**

a. b. 对于“x is 8”，我们需要更有创造性，因为我们需要使用机器指令的交错来找到将x的值确定为9的方法，以便在后续循环中将其评估为8。注意前两个语句块对应于C源代码行，但后面的机器语言语句块交错了源语言语句的一部分。

```
Instruction                              M(x)    P1-R0    P2-R0
P1: LD   R0, x                           10      10       --
P1: DECR R0                              10      9        --
P1: STO  R0, x                           9       9        --

P2: LD   R0, x                           9       9        9
P2: DECR R0                              9       9        8
P2: STO  R0, x                           8       9        8

 P1: LD   R0, x                           8       8        8
 P1: INCR R0                              8       9        --

 P2: LD   R0, x                           8       9        8
 P2: INCR R0                              8       9        9
 P2: STO  R0, x                           9       9        9
 P2: if(x != 10) printf("x is %d", x);
 P2: "x is 9" is printed.

 P1: STO  R0, x                           9       9        9
 P1: if(x != 10) printf("x is %d", x);
 P1: "x is 9" is printed.

 P1: LD   R0, x                           9       9        9
 P1: DECR R0                              9       8        --
 P1: STO  R0, x                           8       8        --

 P2: LD   R0, x                           8       8        8
 P2: DECR R0                              8       8        7
 P2: STO  R0, x                           7       8        7

 P1: LD   R0, x                           7       7        7
 P1: INCR R0                              8       8        7
 P1: STO  R0, x                           8       8        7
 P1: if(x != 10) printf("x is %d", x);
 P1: "x is 8" is printed.
```

5.并发：互斥与同步

- **习题5.8**

平均而言，是的，因为忙等待会消耗无用的指令周期。然而，在特定情况下，如果进程到达程序中必须等待条件满足的某一点，如果该条件已经满足，那么忙等待会立即发现这一点，而阻塞等待将消耗操作系统资源进行进程切换。

5.并发：互斥与同步

- **习题5.24**

所列出的任何交换都会导致程序错误。信号量s控制对临界区的访问，我们只希望临界区包括append或take函数。

5.并发：互斥与同步

- 习题5.25

Scheduled step of execution	full's state & queue	Buffer	empty's state & queue
Initialization	full = 0	OOO	empty = +3
Ca executes c1	full = −1 (Ca)	OOO	empty = +3
Cb executes c1	full = −2 (Ca, Cb)	OOO	empty = +3
Pa executes p1	full = −2 (Ca, Cb)	OOO	empty = +2
Pa executes p2	full = −2 (Ca, Cb)	XOO	empty = +2
Pa executes p3	full = −1 (Cb) Ca	XOO	empty = +2
Ca executes c2	full = −1 (Cb)	OOO	empty = +2
Ca executes c3	full = −1 (Cb)	OOO	empty = +3
Pb executes p1	full = −1 (Cb)	OOO	empty = +2
Pa executes p1	full = −1 (Cb)	OOO	empty = +1
Pa executes p2	full = −1 (Cb)	XOO	empty = +1
Pb executes p2	full = −1 (Cb)	XXO	empty = +1
Pb executes p3	full = 0 (Cb)	XXO	empty = +1
Pc executes p1	full = 0 (Cb)	XXO	empty = 0
Cb executes c2	full = 0	XOO	empty = 0
Pc executes p2	full = 0	XXO	empty = 0
Cb executes c3	full = 0	XXO	empty = +1
Pa executes p3	full = +1	XXO	empty = +1
Pb executes p1-p3	full = +2	XXX	empty = 0
Pc executes p3	full = +3	XXX	empty = 0
Pa executes p1	full = +3	XXX	empty = −1(Pa)
Pd executes p1	full = +3	XXX	Empty = −2(Pa, Pd)
Ca executes c1-c3	full = +2	XXO	empty = −1(Pd) Pa
Pa executes p2	full = +2	XXX	empty = −1(Pd)
Cc executes c1-c2	full = +1	XXO	empty = −1(Pd)
Pa executes p3	full = +2	XXO	empty = −1(Pd)
Cc executes c3	full = +2	XXO	empty =0(Pd
Pd executes p2-p3	full = +3	XXX	empty = 0

5.并发：互斥与同步

- **习题5.24**

对于一个写者和多个读者的代码，如果我们假设读者总是优先的，那么代码是可以的。问题在于读者可能会饿死写者，因为他们可能永远不会全部离开临界区，也就是说，临界区中总是至少有一个读者，因此'wrt'信号量可能永远不会向写者发信号，写者进程也就无法访问'wrt'信号量并写入临界区。

6.并发：死锁与饥饿

- 复习题6.3

  - 互斥，一次只有一个进程可以使用一个资源。
  - 持有并等待，一个进程在等待分配其他资源时，可能会保存已分配的资源。
  - 不可抢占，不能从保存该资源的进程中强制删除任何资源。
  - 循环等待。存在一个封闭的进程链，这样每个进程至少包含链中下一个进程所需的一个资源。

6.并发：死锁与饥饿

- 复习题6.7

  - 死锁预防限制资源请求，以防止至少四种死锁条件中的一种；这可以通过防止三个必要的策略条件之一（互斥、持有和等待、不可抢占）来间接实现，也可以直接通过防止循环等待来实现。
  - 死锁避免允许这三个必要的条件，但做出了明智的选择，以确保不会达到死锁点。通过死锁检测，所请求的资源将尽可能地分配给进程。操作系统周期性地执行一种算法，允许它检测循环等待条件。

6.并发：死锁与饥饿

- 习题6.5
- A.
- 15-(2+0+4+1+1+1)=6
- 6-(0+1+1+0+1+0)=3
- 9-(2+1+0+0+0+1)= 5
- 10-(1+1+2+1+0+1)=4

6.并发：死锁与饥饿

- 习题6.5
- B.需求矩阵 = 最大矩阵 - 分配矩阵

	need			
process	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

6.并发：死锁与饥饿

- 习题6.5
- **C.**下表显示了进程完成的顺序，并显示了每个进程完成后可用的资源

	available			
process	A	B	C	D
P5	7	3	6	5
P4	8	4	6	5
P3	9	4	6	6
P2	13	5	6	8
P1	13	6	7	9
P0	15	6	9	10

6.并发：死锁与饥饿

- 习题6.5
- **D.**答案是不行，原因是如果这个请求被批准，那么新的分配矩阵将会是：

	allocation			
process	A	B	C	D
P0	2	0	2	1
P1	0	1	1	1
P2	4	1	0	2
P3	1	0	0	1
P4	1	1	0	0
P5	4	2	4	4

6.并发：死锁与饥饿

- 习题6.5
- **D.**则新的需求矩阵变为

process	allocation A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	0	2	0	0

6.并发：死锁与饥饿

- 习题6.5
- **D.**可用矩阵变为

Available

A	B	C	D
3	1	2	1

- 这意味着我们不能满足所有进程的需求

6.并发：死锁与饥饿

- 习题6.11
- A.
- 创建该进程将导致以下状态：

Process	Max	Hold	Claim	Free
1	70	45	25	25
2	60	40	20	
3	60	15	45	
4	60	25	35	

- 有足够的可用内存来保证 P1 或 P2 的终止。之后，剩下的三个作业可以按任意顺序完成。

6.并发：死锁与饥饿

- 习题6.11
- **B.**
- 创建该进程将导致导致非常不安全的状态

Process	Max	Hold	Claim	Free
1	70	45	25	15
2	60	40	20	
3	60	15	45	
4	60	35	25	

6.并发：死锁与饥饿

- **习题6.15**

- 确保状态安全所需的可用单元数为 3，使系统中总共有 10 个单元。在问题中显示的状态下，如果再有一个可用单元，P2 可以运行完成，释放其资源，使得 2 个单元可用。这将允许 P1 运行完成，使 3 个单元可用。但此时 P3 需要 6 个单元，P4 需要 5 个单元。如果一开始有 3 个单元可用而不是 1 个单元，那么现在将有 5 个单元可用。这将允许 P4 运行完成，使 7 个单元可用，从而允许 P3 运行完成。

6.并发：死锁与饥饿

- 习题6.18
- A.
- 假设桌子上发生了死锁，即存在一个非空集合 D，其中的每个哲学家 Pi 都持有一只叉子并等待邻居手中的另一只叉子。假设 Pj∈D 是左撇子而不失一般性。由于 Pj 抓着他的左叉子而无法拿到他的右叉子，他的右边邻居 Pk 也永远无法完成他的晚餐，因此 Pk 也是左撇子。因此，Pk ∈ D。继续围绕桌子向右推理，显示 D 中的所有哲学家都是左撇子。这与至少存在一个右撇子的事实相矛盾。因此，不可能发生死锁。

6.并发：死锁与饥饿

- 习题6.18
- B .
  - 假设左撇子 Pj 饿死了，即存在一种稳定的就餐模式，使得 Pj 永远无法进食。假设 Pj 没有持有任何叉子，那么 Pj 的左边邻居 Pi 必须持续持有他的右叉子，并且永远无法完成进餐。因此，Pi 是一个右撇子，持有他的右叉子，但永远无法拿到他的左叉子完成一餐，即 Pi 也会饿死。现在 Pi 的左边邻居也必须是一个右撇子，持续持有他的右叉子。继续围绕桌子向左推理，这表明所有哲学家都是（挨饿的）右撇子。但 Pj 是一个左撇子：这就产生了矛盾。因此，Pj 必须持有一只叉子。
  - 由于 Pj 持续持有一只叉子并等待他的右叉子，Pj 的右边邻居 Pk 永远不会放下他的左叉子，也无法完成一餐，即 Pk 也是一个挨饿的左撇子。如果 Pk 不持续持有他的左叉子，Pj 就可以进食；因此，Pk 持有他的左叉子。继续围绕桌子向右推理，这表明所有哲学家都是（挨饿的）左撇子：这就产生了矛盾。因此，挨饿的情况被排除。

7. 内存管理

- 复习题7.2

通常，程序员无法提前知道在其程序执行时，哪些其他程序会驻留在主内存中。此外，我们希望能够在主内存中交换活跃进程，以通过提供大量就绪进程池来最大化处理器利用率。在这两种情况下，进程在主内存中的具体位置都是不可预测的。

7. 内存管理

- 复习题7.6

内部碎片是指由于加载的数据块小于分区而导致的分区内部浪费的空间。外部碎片是与动态分区相关的现象，是指主内存中大量分区外的小区域累积起来的情况。

7. 内存管理

- 复习题7.7

- 逻辑地址是对内存位置的引用，与当前数据在内存中的分配无关；在进行内存访问之前，必须将其转换为物理地址。相对地址是逻辑地址的一种特定示例，其中地址表示为相对于某个已知点（通常是程序的起始点）的位置。物理地址或绝对地址是主内存中的实际位置。

7. 内存管理

- 复习题7.9

另一种将用户程序细分的方法是分段。在这种情况下，程序及其相关数据被划分为多个段。虽然所有程序的所有段不需要具有相同的长度，但每个段都有一个最大长度。

7. 内存管理

- **习题7.6**
- a. 当 2 MB 的进程被放置时，它填充了所选空闲块的最左侧部分。由于图中在 X 的左侧显示了一个空闲块，因此在 X 放置后交换出的进程必须创建了该空闲块。因此，交换出的进程的最大大小为 1 MB。
- b. 空闲块包括仍然空闲的 5 MB 加上 X 占据的空间，总共为 7 MB。
- c. 答案如下图所示：

7. 内存管理

- **习题7.12**
- a. 逻辑地址空间中的字节数为 2^{16} pages$\times 2^{10}$ bytes/page=2^{26} bytes 。因此，逻辑地址需要 26 位。
- b. 一个帧的大小与一个页的大小相同，为 2^{10} bytes。
- c. 主内存中的帧数为 2^{32} bytes of main memory/2^{10} bytes/frame=2^{22} frames。因此，需要 22 位来指定帧。
- d. 逻辑地址空间中的每个页都有一个条目。因此，共有 2^{16} 个条目。
- e. 除了有效/无效位之外，还需要 22 位来指定主内存中的帧位置，总共需要 23 位。

7. 内存管理

- 习题**7.14**

a.段 0 从位置 660 开始。加上偏移量后，我们有物理地址 660 + 198 = 858。

b. 222 + 156 = 378。

c. 段 1 的长度为 422 字节，所以这个地址会触发段错误。

d. 996 + 444 = 1440。

e. 660 + 222 = 882。

8. 虚拟内存

- **复习题8.1**
- 简单分页：进程的所有页必须在主内存中才能运行，除非使用覆盖。虚拟内存分页：进程的所有页不需要都在主内存中就能运行；页可以根据需要读取。

8. 虚拟内存

- **复习题8.2**
- 抖动是虚拟内存方案中的一种现象，在这种情况下，处理器大部分时间都在交换内存块，而不是执行指令

8. 虚拟内存

- **复习题8.8**
- 时钟策略类似于先进先出 (FIFO) 策略，但在时钟策略中，任何使用位为 1 的帧都会被算法跳过。

8. 内存管理

- 习题8.1

- a. 将二进制地址拆分为虚拟页号（VPN）和偏移量；使用 VPN 作为索引查找页表；提取页帧号（PFN）；连接偏移量以获取物理内存地址。

- b. (i) 1052 = 1024 + 28 映射到 VPN 1 在 PFN 7 中，（7×1024 + 28 = 7196）(ii) 2221 = 2×1024 + 173 映射到 VPN 2，页错误 (iii) 5499 = 5×1024 + 379 映射到 VPN 5 在 PFN 0 中，（0×1024 + 379 = 379）

8. 虚拟内存

- 习题8.4

a. FIFO:

7	0	1	2	0	3	0	4	2	3	0	3	2
7	7	7	2	2	2	2	4	4	4	0	0	0
	0	0	0	0	3	3	3	2	2	2	2	2
		1	1	1	1	0	0	0	3	3	3	3
			F		F	F	F	F	F	F		

8. 虚拟内存

- 习题8.4

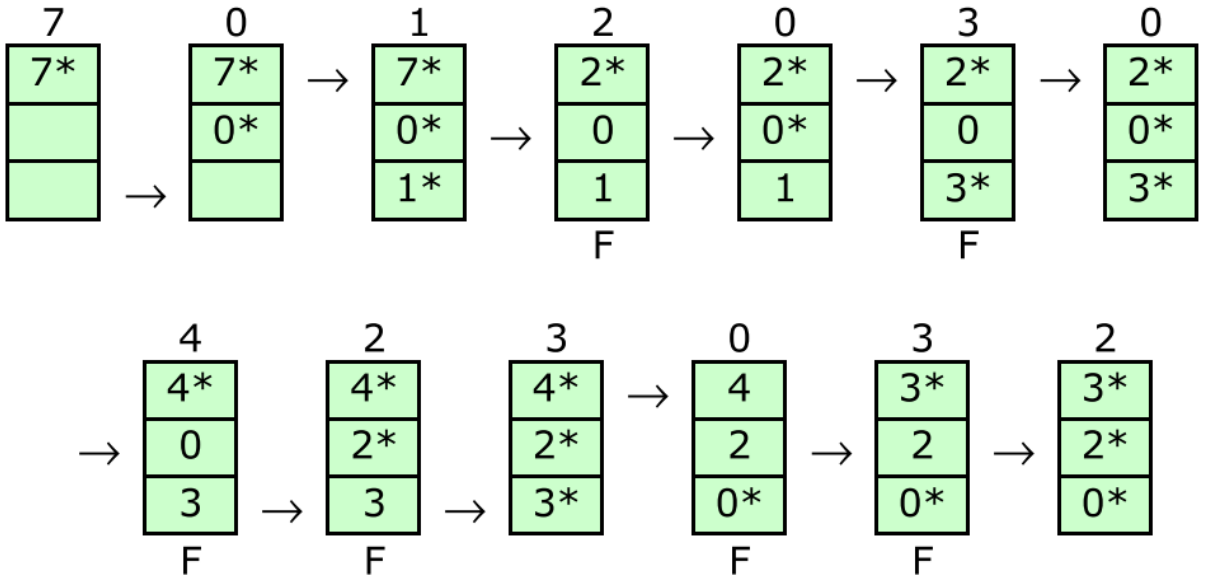
b. LRU:

7	0	1	2	0	3	0	4	2	3	0	3	2
7	7	7	2	2	2	2	4	4	4	0	0	0
	0	0	0	0	0	0	0	0	3	3	3	3
		1	1	1	3	3	3	2	2	2	2	2
			F		F		F	F	F	F		

8. 虚拟内存

- 习题8.4

c. Clock:



8. 虚拟内存

- 习题8.4

d. OPT:

7	0	1	2	0	3	0	4	2	3	0	3	2
7	7	7	2	2	2	2	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4	0	0	0
		1	1	3	3	3	3	3	3	3	3	3
			F		F					F		

8. 虚拟内存

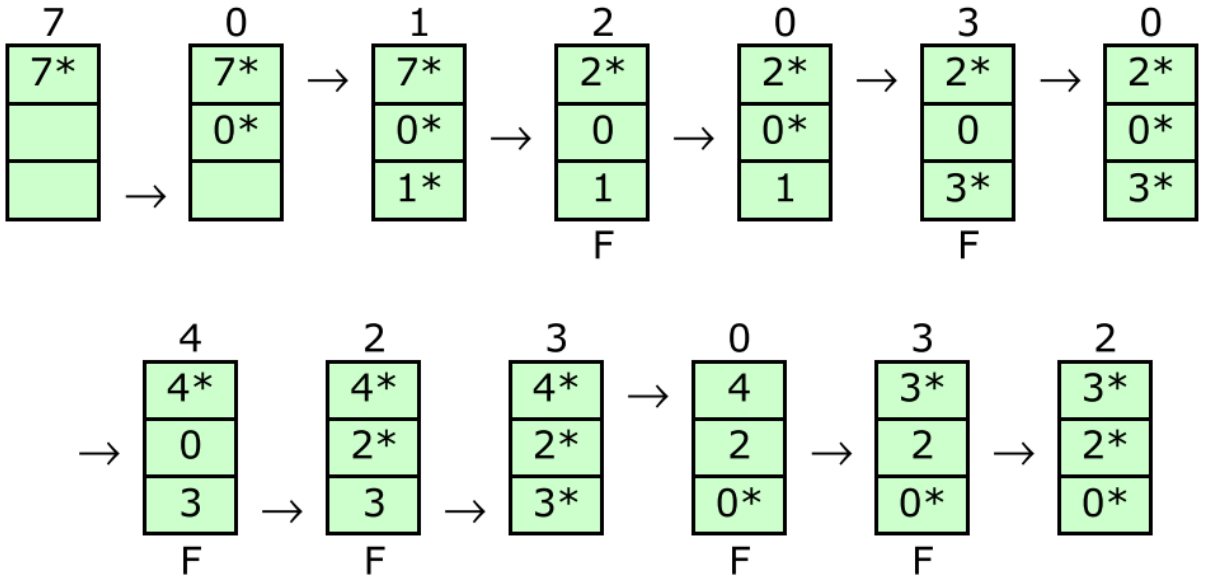
- 习题8.4

FIFO：页错误 = 7，未命中率 = 70%
LRU：页错误 = 6，未命中率 = 60%
Clock：页错误 = 6，未命中率 = 60%
OPT：页错误 = 3，未命中率 = 30%

8.虚拟内存

- 习题8.4

c. Clock:



8.虚拟内存

- 习题8.5

belady现象

8.虚拟内存

- 习题**8.6**

LRU：命中率 = 16/33

1	0	2	2	1	7	6	7	0	1	2	0	3	0	4	5	1	5	2	4	5	6	7	6	7	2	4	2	7	3	3	2	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	2	2	2	2	2	2	2	2
–	0	0	0	0	0	6	6	6	6	2	2	2	2	2	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4
–	–	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7
–	–	–	–	–	7	7	7	7	7	7	7	3	3	3	3	1	1	1	1	1	6	6	6	6	6	6	6	6	3	3	3	3
F	F	F			F	F		F		F		F		F	F	F		F			F	F			F	F			F			

8.虚拟内存

- 习题**8.6**

FIFO:Hit ratio = 16/33

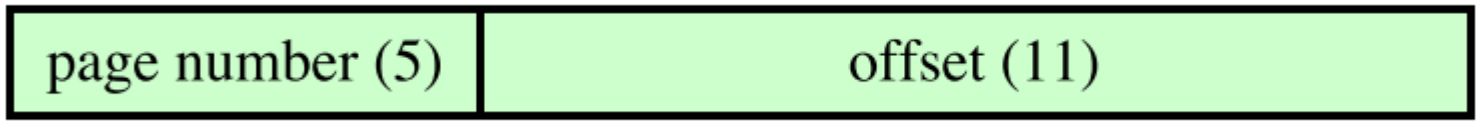
1	0	2	2	1	7	6	7	0	1	2	0	3	0	4	5	1	5	2	4	5	6	7	6	7	2	4	2	7	3	3	2	3
1	1	1	1	1	1	6	6	6	6	6	6	6	6	4	4	4	4	4	4	4	6	6	6	6	6	6	6	6	6	6	2	2
-	0	0	0	0	0	0	0	0	1	1	1	1	1	1	5	5	5	5	5	5	5	7	7	7	7	7	7	7	7	7	7	7
-	-	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
-	-	-	-	-	7	7	7	7	7	7	7	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3
F	F	F			F	F			F		F	F		F	F	F		F			F	F				F			F		F	

对于这个特定的页面访问序列，这两种策略同样有效。

8.虚拟内存

- 习题8.18

a.



b. 32 个条目，每个条目的宽度为 9 位。

c.如果条目总数保持在 32 且页面大小不变，那么每个条目变为 8 位宽

9. 处理机调度

- 复习题9.1

长期调度：决定是否将进程添加到待执行进程池中。

中期调度：决定是否增加部分或全部在主内存中的进程数量。

短期调度：决定处理器将执行哪个可用的进程。

9. 处理机调度

- 复习题9.5

非抢占式：如果一个进程处于运行状态，它将继续执行直到 (a) 它终止或 (b) 它阻塞自己以等待 I/O 或请求某些操作系统服务。

抢占式：当前运行的进程可能会被操作系统中断并移动到就绪状态。抢占的决策可以在新进程到达时、发生中断将阻塞的进程置于就绪状态时，或基于时钟中断定期进行。

9. 处理机调度

- 习题9.2
  - 每个方格代表一个时间单位；方格中的数字表示当前正在运行的进程。

FCFS	A	A	A	B	B	B	B	B	C	C	D	D	D	D	D	E	E	E	E	E
RR, q = 1	A	B	A	B	C	A	B	C	B	D	B	D	E	D	E	D	E	D	E	E
RR, q = 4	A	A	A	B	B	B	B	C	C	B	D	D	D	D	E	E	E	E	D	E
SPN	A	A	A	C	C	B	B	B	B	B	D	D	D	D	D	E	E	E	E	E
SRT	A	A	A	C	C	B	B	B	B	B	D	D	D	D	D	E	E	E	E	E
HRRN	A	A	A	B	B	B	B	B	C	C	D	D	D	D	D	E	E	E	E	E
Feedback, q = 1	A	B	A	C	B	C	A	B	B	D	B	D	E	D	E	D	E	D	E	E
Feedback, q = 2^i	A	B	A	A	C	B	B	C	B	B	D	D	E	D	D	E	E	D	E	E

9. 处理机调度

- 习题9.2

		A	B	C	D	E	
	T_a	0	1	3	9	12	
	T_s	3	5	2	5	5	
FCFS	T_f	3	8	10	15	20	
	T_r	3.00	7.00	7.00	6.00	8.00	6.20
	T_r/T_s	1.00	1.40	3.50	1.20	1.60	1.74
RR $q = 1$	T_f	6.00	11.00	8.00	18.00	20.00	
	T_r	6.00	10.00	5.00	9.00	8.00	7.60
	T_r/T_s	2.00	2.00	2.50	1.80	1.60	1.98
RR $q = 4$	T_f	3.00	10.00	9.00	19.00	20.00	
	T_r	3.00	9.00	6.00	10.00	8.00	7.20
	T_r/T_s	1.00	1.80	3.00	2.00	1.60	1.88
SPN	T_f	3.00	10.00	5.00	15.00	20.00	
	T_r	3.00	9.00	2.00	6.00	8.00	5.60
	T_r/T_s	1.00	1.80	1.00	1.20	1.60	1.32
SRT	T_f	3.00	10.00	5.00	15.00	20.00	
	T_r	3.00	9.00	2.00	6.00	8.00	5.60
	T_r/T_s	1.00	1.80	1.00	1.20	1.60	1.32
HRRN	T_f	3.00	8.00	10.00	15.00	20.00	
	T_r	3.00	7.00	7.00	6.00	8.00	6.20
	T_r/T_s	1.00	1.40	3.50	1.20	1.60	1.74
FB q = 1	T_f	7.00	11.00	6.00	18.00	20.00	
	T_r	7.00	10.00	3.00	9.00	8.00	7.40
	T_r/T_s	2.33	2.00	1.50	1.80	1.60	1.85
FB	T_f	4.00	10.00	8.00	18.00	20.00	
$q = 2^i$	T_r	4.00	9.00	5.00	9.00	8.00	7.00
	T_r/T_s	1.33	1.80	2.50	1.80	1.60	1.81

9. 处理机调度

- **习题9.16**
  - 进程获得 1 分钟处理器时间的顺序如下：

1	2	3	4	5	Elapsed time
A	B	C	D	E	5
A	B	C	D	E	10
A	B	C	D	E	15
A	B		D	E	19
A	B		D	E	23
A	B		D	E	27
A	B			E	30
A	B			E	33
A	B			E	36
A				E	38
A				E	40
A				E	42
A					43
A					44
A					45

  - A = 45 分钟
  - B = 35 分钟
  - C = 13 分钟
  - D = 26 分钟
  - E = 42 分钟
  - 平均周转时间

=45+35+13+26+425=32.2 分钟

9. 处理机调度

- 习题9.16

b.

Priority	Job	Turnaround Time
3	B	9
4	E	9 + 12 = 21
6	A	21 + 15 = 36
7	C	36 + 3 = 39
9	D	39 + 6 = 45

- 平均周转时间为：(9 + 21 + 36 + 39 + 45) / 5 = 30 分钟。

9. 处理机调度

- 习题9.16

c.

Job	Turnaround Time
A	15
B	15 + 9 = 24
C	24 + 3 = 27
D	27 + 6 = 33
E	33 + 12 = 45

- 平均周转时间为: (15 + 24 + 27 + 33 + 45) / 5 = 28.8 分钟

9. 处理机调度

- 习题9.16

  d.

Running Time	Job	Turnaround Time
3	C	3
6	D	3 + 6 = 9
9	B	9 + 9 = 18
12	E	18 + 12 = 30
15	A	30 + 15 = 45

- 平均周转时间为：(3 + 9 + 18 + 30 + 45) / 5 = 21 分钟。

11. I/O 与磁盘调度

- 习题11.3(a)
  - 磁盘磁头最初朝着轨道编号减小的方向移动:

FIFO		SSTF		SCAN		C-SCAN	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
27	73	110	10	64	36	64	36
129	102	120	10	41	23	41	23
110	19	129	9	27	14	27	14
186	76	147	18	10	17	10	17
147	39	186	39	110	100	186	176
41	106	64	122	120	10	147	39
10	31	41	23	129	9	129	18
64	54	27	14	147	18	120	9
120	56	10	17	186	39	110	10
Average	61.8	Average	29.1	Average	29.6	Average	38

12. 文件系统

- **复习题12.3**

文件管理系统是一组系统软件，为用户和应用程序提供使用文件的服务。

12. 文件系统

复习题12.5

文件：数据按到达的顺序收集。每条记录由一组数据组成。

顺序文件：记录使用固定格式。所有记录长度相同，由相同数量的固定长度字段按特定顺序组成。由于每个字段的长度和位置已知，只需存储字段的值；字段名称和长度是文件结构的属性。

索引顺序文件：索引顺序文件保持顺序文件的关键特性：记录按键字段顺序排列。增加了两个特性：文件索引以支持随机访问，以及溢出文件。索引提供快速查找能力以快速到达所需记录的附近。溢出文件类似于顺序文件的日志文件，但集成后通过前驱记录的指针找到溢出文件中的记录。

索引文件：记录仅通过其索引访问。因此，只要至少一个索引中的指针指向该记录，记录的位置就没有限制。此外，可以使用变长记录。

直接文件或哈希文件：直接文件使用键值的哈希。

12. 文件系统

- 复习题12.7

搜索、创建文件、删除文件、列出目录、更新目录

12. 文件系统

- **复习题12.11**

连续分配：在文件创建时，为文件分配一组连续的块。

链式分配：按单个块分配。每个块包含一个指向链中下一个块的指针。

索引分配：文件分配表为每个文件包含一个独立的一级索引；索引对文件分配的每个部分都有一个条目。

12. 文件系统

- 习题12.12

Level	Number of Blocks	Number of Bytes
Direct	10	10K
Single indirect	256	256K
Double indirect	256 × 256 = 65K	65M
Triple indirect	256 × 65K = 64M	16G

最大文件大小超过 16G 字节。