

操作系统课后作业答疑

Presented by Zhangyuan, Zhaokangming

SCUT Operating System



Jun 15, 2021

Contents

1 Homework

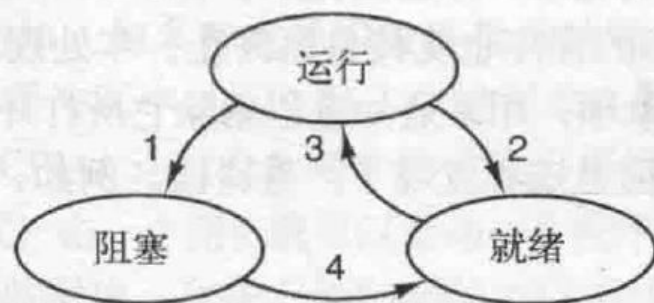
2 F&Q

Contents

- 1 Homework** 
 - **Ch2** 1、 4、 14、 18、 29、 34、 39、 45、 60
 - **Ch6** 23、 26、 31、 34、 41
- 2 F&Q** 
 - **Ch3**
4、 6、 7、 17、 19、 25、 28、 30、 32、 36、 38

Ch2.1

1. 图2-2中给出了三个进程状态。理论上，三个状态之间可以有六种转换，每个状态两个。但图中只给出了四种转换。其余两种转换是否可能发生？

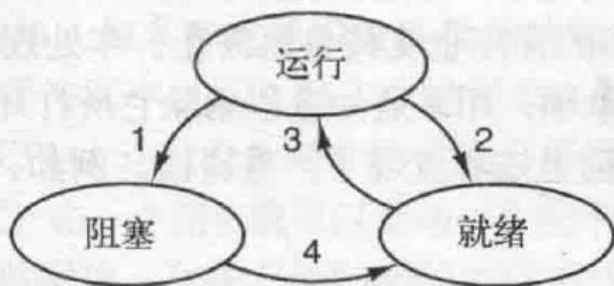


1. 进程因为等待输入而被阻塞
2. 调度程序选择另一个进程
3. 调度程序选择这个进程
4. 出现有效输入

图2-2 一个进程可处于运行态、阻塞态和就绪态，图中显示出各状态之间的转换

■ Answer : 阻塞→运行: 可能发生 就绪→阻塞: 不可能发生

Ch2.1



1. 进程因为等待输入而被阻塞
2. 调度程序选择另一个进程
3. 调度程序选择这个进程
4. 出现有效输入

- 1) 运行态 (该时刻进程实际占用CPU)。
- 2) 就绪态 (可运行, 但因为其他进程正在运行而暂时停止)。
- 3) 阻塞态 (除非某种外部事件发生, 否则进程不能运行)。

当一个进程在逻辑上不能继续运行时, 它就会被阻塞, 典型的例子是它在等待可以使用的输入。

■ Answer :

阻塞→运行: 可能发生, 例如在I/O上发生阻塞, 当I/O满足后就可以继续运行

就绪→阻塞: 不可能发生

Ch2.4

4. 中断或系统调用把控制权转交给操作系统时，为什么通常会用到与被中断进程的栈分离的内核栈？

■ 本题实际是问**为什么 进程栈 与 内核栈 分离？**

■ Answer :

为内核使用单独的堆栈有几个原因。其中两个如下。

- 不希望操作系统崩溃，因为编写得不好的用户程序不允许有足够的堆栈空间
- 如果内核在从系统调用返回时将堆栈数据留在用户程序的内存空间中，恶意用户可能能够使用这些数据来查找有关其他进程的信息。

Ch2.14

14. 既然计算机中只有一套寄存器，为什么图2-12中的寄存器集合是按每个线程列出而不是按每个进程列出？

| 每个进程中的内容 | 每个线程中的内容 |
|-----------|----------|
| 地址空间 | 程序计数器 |
| 全局变量 | 寄存器 |
| 打开文件 | 堆栈 |
| 子进程 | 状态 |
| 即将发生的定时器 | |
| 信号与信号处理程序 | |
| 账户信息 | |

图2-12 第一列给出了在一个进程中所有线程共享的内容，第二列给出了每个线程自己的内容

■ **Answer :** 进程和线程之间：一个进程可以拥有多个线程，当一个线程停止时，它在寄存器中有值，那么就必须被保存，所以每个线程需要自己的寄存器保存区而非按照线程的层次保存。当然进程停止时，也必须其保存寄存器。

18. 在用户态实现线程的最大优点是什么？最大缺点是什么？

■ Answer :

最大的优点是效率高，不需要借助内核来切换线程。

最大的缺点是，用户态线程如果一个线程阻塞，整个进程都会阻塞。

补充：协程，英文Coroutines，是一种基于线程之上，但又比线程更加轻量级的存在，这种由程序员自己写程序来管理的轻量级线程叫做『用户空间线程』，具有对内核来说不可见的特性。因为是自主开辟的异步任务，所以很多人也更喜欢叫它们纤程（Fiber），或者绿色线程（GreenThread）。

Ch2.29

29. 将生产者-消费者问题扩展成一个多生产者-多消费者的问题，生产（消费）者都写（读）一个共享的缓冲区，每个生产者和消费者都在自己的线程中执行。图2-28中使用信号量的解法在这个系统中还可行吗？

```
#define N 100                                     /* 缓冲区中的槽数目 */
typedef int semaphore;                          /* 信号量是一种特殊的整型数据 */
semaphore mutex = 1;                            /* 控制对临界区的访问 */
semaphore empty = N;                            /* 计数缓冲区的空槽数目 */
semaphore full = 0;                             /* 计数缓冲区的满槽数目 */

void producer(void)
{
    int item;

    while (TRUE) {
        item = produce_item();                /* TRUE是常量1 */
        down(&empty);                          /* 产生放在缓冲区中的一些数据 */
        down(&mutex);                          /* 将空槽数目减1 */
        insert_item(item);                    /* 进入临界区 */
        up(&mutex);                            /* 将新数据项放到缓冲区中 */
        up(&full);                             /* 离开临界区 */
        up(&full);                             /* 将满槽的数目加1 */
    }
}

void consumer(void)
{
    int item;

    while (TRUE) {
        down(&full);                          /* 无限循环 */
        down(&mutex);                          /* 将满槽数目减1 */
        item = remove_item();                 /* 进入临界区 */
        up(&mutex);                            /* 从缓冲区中取出数据项 */
        up(&empty);                            /* 离开临界区 */
        consume_item(item);                  /* 将空槽数目加1 */
    }
}
```

图2-28 使用信号量的生产者-消费者问题

■ **Answer:**可行。

在给定的时刻，只有一个生产者（消费者）可以向缓冲区添加（移除）一个项目。

34. 如果线程在内核态实现，可以使用内核信号量对同一个进程中的两个线程进行同步吗？如果线程在用户态实现呢？假设其他进程中没有线程需要访问该信号量。请解释你的答案。

■ **Answer :**

对于内核线程，线程可以在信号量上阻塞，而内核可以运行该进程中的其它线程。因而，使用信号量没有问题。而对于用户级线程，当某个线程在信号量上阻塞时，内核将认为整个进程都被阻塞，而且不再执行它。因此，进程失败。

- **补充:** 线程间通信方式包括临界区，互斥量，信号量，条件变量，事件。其中信号量为控制具有有限数量的用户资源而设计的，它允许多个线程在同一时刻去访问同一个资源，但一般需要限制同一时刻访问此资源的最大线程数目。信号量是一种特殊的变量，可用于线程同步。它只取自然数值，并且只支持两种操作 $P(SV)$, $V(SV)$ 。

Ch2.39

39. 考虑以下C代码：

```
void main() {  
    fork();  
    fork();  
    exit();  
}
```

程序执行时创建了多少子进程？

- Answer : 3个。
- 注意审题：子进程。
- 补充：父进程和子进程共享代码段，在fork后分叉，分别执行各自的后续代码。

45. 有5个批处理作业A~E，它们几乎同时到达一个计算中心。估计它们的运行时间分别为10、6、2、4和8分钟。其优先级（由外部设定）分别为3、5、2、1和4，其中5为最高优先级。对于下列每种调度算法，计算其平均进程周转时间，可忽略进程切换的开销。

(a) 轮转法

(b) 优先级调度

(c) 先来先服务（按照10、6、2、4、8次序运行）

(d) 最短作业优先

对于 (a)，假设系统具有多道程序处理能力，每个作业均公平共享CPU时间；对于 (b)~(d)，假设任一时刻只有一个作业运行，直到结束。所有的作业都是CPU密集型作业。

Ch2.45

■ Answer :

| 作业名称 | A | B | C | D | E |
|------|----|---|---|---|---|
| 运行时间 | 10 | 6 | 2 | 4 | 8 |
| 优先级 | 3 | 5 | 2 | 1 | 4 |

完成时间:

| | | | | | | AVG |
|------------|----|----|----|----|----|------|
| a) 轮转法。 | 30 | 24 | 10 | 18 | 28 | 22 |
| b) 优先级调度。 | 24 | 6 | 26 | 30 | 14 | 20 |
| c) 先来先服务。 | 10 | 16 | 18 | 22 | 30 | 19.2 |
| d) 最短作业优先。 | 30 | 12 | 2 | 6 | 20 | 14 |

Ch2.60

学校颁布法令：当有一个女生在浴室里时，其他女生可以进入，但是男生不行，反之亦然。在每个浴室的门上有一个滑动标记，表示当前处于以下三种可能状态之一：

- 空
- 有女生
- 有男生

用你喜欢的程序设计语言编写下面的过程：
woman_wants_to_enter, man_wants_to_enter,
woman_leaves, man_leaves。可以随意使用计数器
和同步技术。

■ **Answer：**代码后续放在仓库中。仅供参考。

提示：信号量，条件变量。

Ch6.23

22. 某一系统有两个进程和三个相同的资源。每个进程最多需要两个资源。这种情况下有没有可能发生死锁？为什么？
23. 重新考虑上一题，假设现在共有 p 个进程，每个进程最多需要 m 个资源，并且有 r 个资源可用。什么样的条件可以保证死锁不会发生？

■ Answer 23: $r \geq p(m-1) + 1$

Ch6.26

26. 一个系统有4个进程和5个可分配资源，当前分配和最大需求如下：

| | 已分配资源 | 最大需求量 | 可用资源 |
|-----|-----------|-----------|-----------|
| 进程A | 1 0 2 1 1 | 1 1 2 1 3 | 0 0 × 1 1 |
| 进程B | 2 0 1 1 0 | 2 2 2 1 0 | |
| 进程C | 1 1 0 1 0 | 2 1 3 1 0 | |
| 进程D | 1 1 1 1 0 | 1 1 2 2 1 | |

若保持该状态是安全状态，x的最小值是多少？

■ Answer : Min $x = 2$

Ch6.26

26. 一个系统有4个进程和5个可分配资源，当前分配和最大需求如下：

| | 已分配资源 | 最大需求量 | 可用资源 |
|-----|-----------|-----------|-----------|
| 进程A | 1 0 2 1 1 | 1 1 2 1 3 | 0 0 × 1 1 |
| 进程B | 2 0 1 1 0 | 2 2 2 1 0 | |
| 进程C | 1 1 0 1 0 | 2 1 3 1 0 | |
| 进程D | 1 1 1 1 0 | 1 1 2 2 1 | |

若保持该状态是安全状态，x的最小值是多少？

■ Answer : Min $x = 2$

31. 一种预防死锁的方法是去除占有和等待条件。在本书中，假设在请求一个新的资源以前，进程必须释放所有它已经占有的资源（假设这是可能的）。然而，这种做法会引入这样的危险性：竞争的进程得到了新的资源但却丢失了原有的资源。请给出这一方法的改进。

- **Answer :** 题目中请求新资源前要释放占用的资源这明显是不合理的。改进：进程先保留自己已经占有的资源，再请求新的资源，若能满足则使用完成后释放；如果不能或者资源不可用，则释放所有资源。

34. 解释死锁、活锁和饥饿的区别。

- **Answer :**
- **死锁：**当一组进程被阻塞，等待只有该集合中的其他进程可以释放的资源时，会发生死锁。
- **活锁：**活锁中的进程不会被阻塞，它们会继续检查永远不会成立的执行条件是否成立。
- **饥饿：**饥饿是由于存在其他进程以及新的进程流，这些进程的优先级比饥饿的进程高，最终会导致饥饿的进程无限期的推迟。

41. 编写银行家算法的模拟程序。该程序应该能够循环检查每一个提出请求的银行客户，并且能判断这一请求是否安全。请把有关请求和相应决定的列表输出到一个文件中。

- **Answer :**代码后续放在仓库中。仅供参考。
- **提示：** 循环检查，安全判定标准。

4. 在一个交换系统中，按内存地址排列的空闲区大小是10MB、4MB、20MB、18MB、7MB、9MB、12MB和15MB。对于连续的段请求：

(a) 12MB

(b) 10MB

(c) 9MB

使用首次适配算法，将找出哪个空闲区？使用最佳适配、最差适配、下次适配算法呢？

第一次安装需要20 MB、10 MB、18 MB。最佳匹配需要12 MB、10 MB和9 MB。最差的匹配需要20 MB、18 MB和15 MB。下一个匹配需要20 MB，18MB和9MB。

6. 对下面的每个十进制虚拟地址，分别使用4KB页面和8KB页面计算虚拟页号和偏移量：20000，32768，60000。

对于4kb的页面大小，(page, offset) 对是 (4, 3616) , (8, 0) 和 (14, 2656) 。对于8kb的页面大小，它们是 (2, 3616) 、 (4, 0) 和 (7, 2656) 。

注：回答下面的也算正确

对于4kb的页面大小，(page, offset) 对是 (3, 3616) , (7, 0) 和 (13, 2656) 。对于8kb的页面大小，它们是 (1, 3616) 、 (3, 0) 和 (6, 2656) 。

Ch 3.7

7. 使用图3-9的页表，给出下面每个虚拟地址对应的物理地址：

- (a) 20
- (b) 4100
- (c) 8300

7、 (a) 8212. (b) 4100. © 24684.

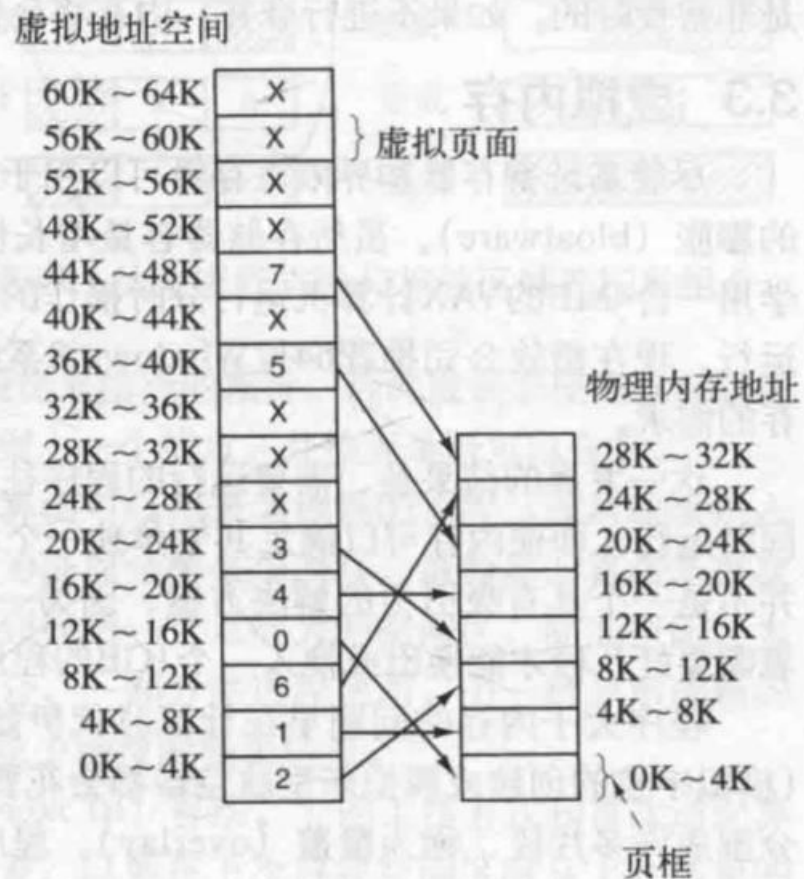


图3-9 页表给出虚拟地址与物理内存地址之间的映射关系。每一页起始于4096的倍数位置，结束于起址加4095的位置，所以4K到8K实际为4096~8191，8K到12K就是8192~12 287

Ch 3.17

17. 假设一个机器有38位的虚拟地址和32位的物理地址。

(a) 与一级页表比较，多级页表的主要优点是什么？

(b) 若采用二级页表，页面大小为16KB，每个页表项为4字节，应该对第一级页表域分配多少位？对第二级页表域分配多少位？请解释原因。

17、考虑一下，

(A) 多级页表减少了由于其层次结构而需要在内存中的页表的实际页数。事实上，在一个具有大量指令和数据位置的程序中，我们只需要顶层页表（一页）、一个指令页和一个数据页。

(b) 为三页字段中的每个字段分配12位。“偏移”字段需要

14位到地址16KB。它为页字段留下24位。因为每个

条目为4字节，一页可容纳2个12页的表条目，因此-

需要12位来索引一页。所以为每个页面分配12位字段将寻址所有2个38字节。

Ch 3.19

19. 一个32位地址的计算机使用两级页表。虚拟地址被分成9位的第一级页表域、11位的二级页表域和一个偏移量，页面大小是多少？在地址空间中一共有多少个页面？

19、20位用于虚拟页码，剩下12位用于偏移。这将生成一个4-kb的页面。虚拟页的20位意味着 2^{20} 次方页。

25. 一个计算机的页面大小为8KB，主存大小为256KB，64GB虚拟地址空间使用倒排页表实现虚拟内存。为了保证平均散列链的长度小于1，散列表应该多大？假设散列表的大小为2的幂。

25、主存储器有 $2^{28}/2^{13}=2^{15}=32768$ 页。
32K哈希表的平均链长度为1。要低于1，我们必须转到下一个大小，65536个条目。将32768个条目分布在65536个表槽上将得到0.5的平均链长度，从而确保快速查找。

Ch 3.28

28. 如果将FIFO页面置换算法用到4个页框和8个页面上，若初始时页框为空，访问序列串为0172327103，请问会发生多少次缺页中断？如果使用LRU算法呢？

● FIFO的page frames如下：

1. x0172333300
2. xx017222233
3. xxx01777722
4. xxxx0111177

FIFO发生6次缺页，LRU为7次

● LRU的page frames如下：

1. x0172327103
2. xx017232710
3. xxx01773271
4. xxxx0111327

Ch 3.30

30. 一台小计算机有4个页框。在第一个时钟周期时R位是0111（页面0是0，其他页面是1），在随后的时钟周期中这个值是1011、1010、1101、0010、1010、1100、0001。如果使用带有8位计数器的老化算法，给出最后一个时钟周期后4个计数器的值。

| | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|
| Page 0: | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| Page 1: | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Page 2: | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Page 3: | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Ch 3.32

32. 在图3-20 c的工作集时钟算法中，表针指向那个 $R=0$ 的页面。如果 $\tau=400$ ，这个页面将被移出吗？如果 $\tau=1000$ 呢？

32、页面的年龄为 $2204-1213=991$ 。如果 $\tau=400$ ，它肯定不在工作集内，并且最近没有被引用，因此它将被逐出。 $\tau=1000$ 情况不同。现在，页面属于工作集（几乎不属于），因此不会被删除。

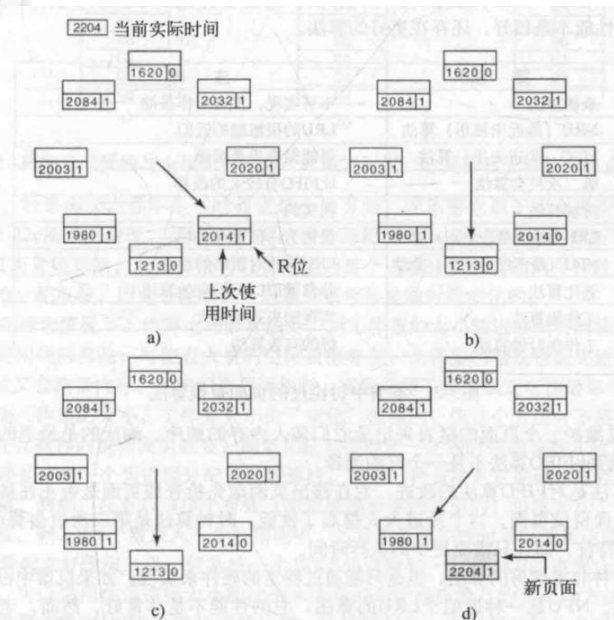


图3-20 工作集时钟页面置换算法的操作：a)和b)给出在 $R=1$ 时所发生的情形；c)和d)给出 $R=0$ 的例子

Ch 3.36

36. 一个计算机有4个页框，载入时间、最近一次访问时间和每个页的R位和M位如下所示（时间以一个时钟周期为单位）：

| 页面 | 载入时间 | 最近一次访问时间 | R | M |
|----|------|----------|---|---|
| 0 | 126 | 280 | 1 | 0 |
| 1 | 230 | 265 | 0 | 1 |
| 2 | 140 | 270 | 0 | 0 |
| 3 | 110 | 285 | 1 | 1 |

- (a) NRU算法将置换哪个页面？
- (b) FIFO算法将置换哪个页面？
- (c) LRU算法将置换哪个页面？
- (d) 第二次机会算法将置换哪个页面？

36、
NRU删除第2页。
FIFO删除第3页。
LRU删除第1页。
第二次机会删除第2页。

Ch 3.38

38. 有二维数组：

```
int X[64][64];
```

假设系统中有4个页框，每个页框大小为128个字（一个整数占用一个字）。处理数组X的程序正好可以放在一页中，而且总是占用0号页。数据会在其他3个页框中被换入或换出。数组X为按行存储（即，在内存中，X[0][0]之后是X[0][1]）。下面两段代码中，哪一个会有最少的缺页中断？请解释原因，并计算缺页中断的总数。

A段

```
for (int j=0; j<64; j++)  
    for (int i=0; i<64; i++) X[i][j]=0;
```

B段

```
for (int i=0; i<64; i++)  
    for (int j=0; j<64; j++) X[i][j]=0;
```

38、片段B是因为代码比片段A具有更多的空间位置。对于外部循环的其他迭代，内部循环只会导致一个页面错误。（只有32个页面错误。）除此之外（片段A）：由于一个帧是128个字，x数组的一行占据了一页的一半（即64个字）。整个数组适合 $64 \times 32 / 128 = 16$ 个帧。对于给定的列，代码的内部循环将逐步通过连续的x行。因此，其他对x[i][j]的引用都会导致页面错误。页面错误总数为 $64 \times 64 / 2 = 2048$ 。

Contents

1 Homework

2 F&Q

Thank You

如果后续还有任何问题，请到仓库

<https://github.com/zhangyuanes/OperatingSystemExperiments>中去提交issue。

Thank You

Q&A