

CPU 调度：

1. 调度的基本概念

处理机调度负责动态地把处理器分配给进程或内核级线程。处理机调度也称CPU调度或进程调度，在有线程的操作系统中也称线程调度。为了最大限度的提高CPU利用率，多道程序设计的目标是保持总是有进程可供执行。在单处理机系统中，一次只能运行一个进程；其它的任何进程都必须等到CPU空闲时才能够被重新调度。

多道程序设计的思想十分简单。一个进程持续运行直到它必须等待某些操作（I/O请求是个典型）的完成。在简单的计算机系统中，进程等待时CPU将处于空闲状态；这浪费了所有的等待时间。利用多道程序设计，我们可以有效地利用这段时间。在内存中同时保留多个进程。当一个进程必须等待时，操作系统将CPU撤离该进程并把CPU分配给另一个进程。然后以这种方式继续运行。

从处理器调度的对象、时间、功能等不同角度，我们可把处理器调度分成不同类型。处理器调度不仅涉及选择哪一个就绪进程进入运行状态，还涉及何时启动一个进程的执行。按照调度所涉及的层次的不同，我们可把处理器调度分成高级调度、中级调度和低级调度三个层次。

高级调度也称为作业调度或宏观调度。从用户工作流程的角度，一次作业提交若干个流程，其中每个程序按照流程进行调度执行。中级调度涉及进程在内外存间的交换。从存储器资源管理的角度来看，把进程的部分或全部换出到外存上，可为当前运行进程的执行提供所需的内存空间，将当前进程所需部分换入到内存。指令和数据必须在内存里才能被处理器直接访问。低级调度也称为微观调度。从处理器资源分配的角度来看，处理器需要经常选择就绪进程或线程进入运行状态。

2. 调度时机、切换与过程

有四种情况都会发生CPU调度：当一个进程从运行状态转换到等待状态时；当一个进程从运行状态转换到就绪状态时；当一个进程从等待状态转换到就绪状态时；当一个进程终止运行时。

从操作系统角度观察，进程或者其PCB总是在各种等待队列中，或者在队列之间迁移。如图2-8。

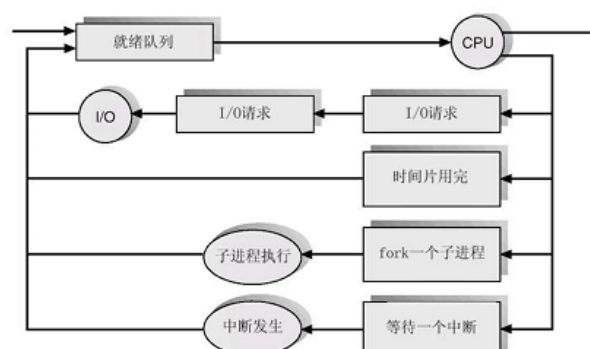


图 2-8 进程在各队列中的迁移

其中，就绪队列是个特殊的队列，它所包含的进程都已经得到了几乎所有资源，只缺少CPU了。CPU调度的任务，就是从就绪队列中选择一个等待进程，并为其分配CPU；选

择的目标，则是如何使 CPU 得到最好的利用，用户进程得到最好的服务。从图中看，CPU 调度使选中的进程从“就绪队列”节点，迁移到“CPU”节点。操作系统配备有专门的调度程序，列入 schedule() 函数，只有该调度程序被调用执行了，才会发生一次 CPU 调度。所谓调度时机，就是什么时候去调用这个程序。

3. 调度的基本准则

在操作系统中，如何选择调度方式和算法，在很大程度上取决于操作系统的类型和目标，选择调度方式和算法的准则有的是面向用户的，有的是面向系统的。常见的指标有 CPU 利用率、吞吐率、响应时间、周转时间、等待时间等。

(1) 面向用户的准则和评价

1) 周转时间短

周转时间短是评价批处理系统的重要性能指标。作业周转时间是指从作业提交给系统开始，到作业完成为止的这段时间间隔。

计算机系统要使大多数用户对周转时间感到满意，引入平均周转时间 T 的评价指标，对有 n 个作业的系统 T 定义为

$$T = 1/n \times \left[\sum_{i=1}^n T_i \right]$$

T_i 为第 i 个作业的周转时间。

2) 响应时间快

响应时间是评价分时系统的性能指标。响应时间是从用户通过键盘提交一个请求开始，直至系统首次产生响应为止的时间。

3) 截止时间的保证

它是用来评价实时系统的重要指标，截止时间是某任务必须执行的最迟时间，或完成的最迟时间。

4) 优先权准则

在选择批处理、分时和实时系统的调度算法时，都可引用优先权准则，以便让那些紧急的作业（或事件），得到及时的处理。在要求较严格的场合，往往还需选择抢占调度方式，才能保证紧急作业得到及时的处理。

(2) 面向系统的准则

1) 达到系统设计目标

系统的设计目标是选择算法的主要依据。例如批处理系统所追求的是充分发挥和提高计算机的效率，分时系统则侧重于保护用户的请求及时给予响应，实时系统所关心的是不要丢失实时信息并给予处理，计算中心要求系统吞吐量要大等。

2) 系统吞吐量大

用来评价批处理系统的重要指标。系统吞吐量是单位时间内完成的作业数，它与批处理作业的平均长度具有密切关系。

3) 处理机利用率高

对于大中型多用户系统，由于 CPU 价格十分昂贵，所以处理机利用率成为衡量大、中型系统性能的十分重要指标，但对单用户微机或某些实时系统，该准则就不那么重要。

4) 各类资源的平衡利用

在中大型系统中，有效地利用各类资源（包括 CPU、外存、I/O 设备等）也是一个重要指标，对于微型机和某些实时系统，该准则也不重要。

4.调度方式

进程调度可采用下述两种方式：

(1) 非抢占方式

采用这种调度方式时，一旦把处理机分配给某进程后，便让进程一直执行，直到该进程完成或发生某事件而被阻塞时，才把处理机分配给其它进程，不允许某进程抢占已经分配出去的处理机。

(2) 抢占方式

这种调度方式，允许进程调度程序根据某个原则，去停止某个正在执行的进程，将已分配给进程的处理机，重新分配给另一个进程。抢占的原则有：

- 时间片原则。各进程按时间片运行，当一个时间片用完后，便停止该进程的执行而重新进行调度。这个原则适用于分时系统。
- 优先权原则。通常是对一些重要的和紧急的进程赋予较高的优先权。当这种进程进入就绪队列时，例如由阻塞态转换为就绪态，或从静止就绪态转为活动就绪态时，或新创建进入就绪态的进程进入就绪队列时，如果其优先权比正在执行的进程优先权高，便停止正在执行的进程，将处理机分配给优先权高的进程，使之执行。

5.典型调度算法

典型的调度算法有：先来先服务调度算法，短作业（短任务、短进程、短线程）优先调度算法，时间片轮转调度算法，优先级调度算法，高响应比优先调度算法，多级反馈队列调度算法，等等。

先来先服务（First Come First Served, FCFS）调度算法是按照进程进入就绪队列的先后次序来挑选进程，先进入就绪队列的进程优先被挑中。先来先服务调度算法是最简单的调度算法，但是它会让短进程等待非常长的进程。FCFS 会产生所谓的 Belady 异常现象。

最短作业/进程优先（Shortest Job/Process First, SJF/SPF）调度算法是以进程所要求的 CPU 时间为标准，总是选取估计计算时间最短的进程投入运行。最短进程优先调度算法是局部最佳的方法，它满足最短平均等待时间的要求。但实现 SJF 调度比较困难，因为预测进程的下一个 CPU 需求区间的长度有难度。SJF 算法是优先权调度算法的特例，优先权调度和 SJF 调度会产生饥饿，老化（Aging）技术可解决饥饿问题。

时间片轮转（Round Robin, RR）调度算法是调度程序每次把 CPU 分配给就绪队列首进程使用一个时间片，例如 100ms，就绪队列中的每个进程轮流地运行一个这样的时间片。时间片轮转调度算法对于分时（交互）系统更为合适。RR 算法的主要问题是选择时间片，如果时间片太大，那么 RR 调度就成了 FCFS 调度；如果时间片太小，那么因上下文切换而引起的调度开销就过大。

优先级（Priority）调度算法是根据确定的优先数来选取进程，每次总是选择优先级高的进程。规定用户进程优先数的方法是多种多样的，进程的优先级的设置可以是静态的，也可以是动态的。

高响应比优先（Highest Response Ratio Next, HRRN）调度算法计算就绪队列进程的响应，调度时总是选择响应比最高的就绪进程得到 CPU。响应比 = (进程已等待时间 + 进程要求运行时间) / 进程要求运行时间。此调度算法首先有利于短进程，但也兼顾到等待时间长的进程，该算法是 FCFS 算法和 SJF 算法的折衷。

多队列调度算法（Multilevel Queue, MQ）是根据进程的性质和类型的不同，将就绪队列再分为若干个子队列，所有进程根据不同情况排入相应的队列中，而不同的就绪队列采用不同的调度算法。最为常用的是前台交互队列（使用 RR 调度）和后台批处理队列（使用 FCFS 调度）的组合。

多级反馈队列（Multilevel Feedback Queue Scheduling, MFQ）调度算法在多级队列算法的基础上，允许就绪进程在队列之间迁移。

FCFS 算法是非抢占的，RR 算法是抢占的，SJF 算法和优先级算法既可以是抢占的，也可以是非抢占的。

如果操作系统在内核级支持线程，那么必须调度线程而不是调度进程。

6. 多线程程序实际编程注意事项:

//创建线程

```
pthread_create(&tid, &attr, runner, NULL);
```

```
pthread_join(tid, NULL);
```

在 Ubuntu 下面编译多线程程序命令

(gcc example.c -o yourname -lpthread)及避免连续的 `execlp()`（注意：该命令完全替换当前进程的代码段、数据段和堆栈，原进程的代码从调用点起被新程序覆盖。如果子进程执行两个 `execlp` 调用，第二个 `execlp` 调用将不会被执行。）；若需执行多个命令，应使用 `system()` 或多次创建子进程。

进程同步：

3.5.1 有限缓冲区问题 (bounded-buffer problem, 生产者消费者问题)

有限缓冲区问题先前已讨论过，它通常用来说明同步原语的能力。这里，介绍一种该方案的通用解决结构，而不是只局限于某个特定实现。

假定缓冲池有 n 个缓冲项，每个缓冲项能存放一个数据项。二进制信号量 `mutex` 提供了对缓冲池访问的互斥要求，并初始化为 1。计数信号量 `empty` 和 `full` 分别用来表示空缓冲项和满缓冲项的个数。信号量 `empty` 初始化为 n ；而信号量 `full` 初始化为 0。

生产者进程的代码如图 3.14 所示；消费者进程的代码如图 3.15 所示。注意生产者和消费者之间的对称性。可以这样来理解代码：生产者为消费者生产满缓冲项，而消费者为生产者生产空缓冲项。

```
do {
    // produce an item in nextp
    wait(empty);
    wait(mutex);
    // add nextp to buffer
    signal(mutex);
    signal(full);
} while (TRUE);
```

图 3.14 生产者进程结构

```
do {
    wait(full);
    wait(mutex);
    // remove an item from buffer to nextc
    signal(mutex);
    signal(empty);
    // consume the item in nextc
} while (TRUE);
```

图 3.15 消费者进程结构

3.5.2 读者—写者问题 (Readers-writers problem)

一个数据库正被多个并发进程所共享。其中，有的进程可能只需要读数据库，而其他进程可能需要更新（即读和写）数据库。为了区分这两种类型的进程，将前者称为读者，而将后者称为写者。显然，如果两个读者同时访问共享数据，那么不会产生什么不利的结果。然而，如果一个写者和其他进程（既不是读者也不是写者）同时访问共享对象，很可能引起混乱。

为了确保不会产生这样的混乱，要求写者对共享数据库有排他的访问。这一同步问题称为读者—写者问题。自从它被提出后，就一直用来测试几乎所有新的同步原语。

读者—写者问题有多个变种，都与优先级有关。最为简单的，通常也被称为第一类读者—写者问题，要求除非已有一个写者已获得允许以使用共享数据库，否则没有读者需要保持等待。换句话说，没有读者会因为有一个写者在等待而会等待其他读者的完成。第二类读者—写者问题则要求，

一旦写者就绪，那么写者会尽可能快地执行其写操作。换句话说，如果一个写者等待访问数据库，那么不会有新读者开始读操作。

对这两个问题的解答都可能导致饥饿问题。对第一种情况，写者可能饥饿；对第二种情况，读者可能饥饿。由于这个原因，已经提出了问题的其他变种。

这里介绍一个对第一类读者—写者问题的解答。关于读者—写者问题的没有饥饿的解答，请参见推荐的有关文献。

对于第一读者—写者问题的解决，读者进程共享以下数据结构：

```
semaphore mutex, wrt;  
int readcount;
```

信号量 mutex 和 wrt 初始化为 1；readcount 初始化为 0。信号量 wrt 为读者和写者进程所共用。信号量 mutex 用于确保在更新变量 readcount 时的互斥。变量 readcount 用来跟踪有多少进程正在读对象。信号量 wrt 供写者作为互斥信号量。它为第一个进入临界区和最后一个离开临界区的读者所使用，而不被其他读者所使用。

写者进程的代码如图 3.16 所示，读者进程的代码如图 3.17 所示。注意，如果有一个写者进程在临界区内，且 n 个读者进程处于等待，那么一个读者在 wrt 上等待，而 n-1 个读者在 mutex 上等待。而且，当一个写者执行 signal(wrt)时，可以重新启动等待读者或写者的执行。这一选择由调度程序所做。

```
do {  
    wait(wrt);  
    // writing is performed  
    signal(wrt);  
} while (TRUE);
```

图 3.16 写者进程的结构

```
do{  
    wait(mutex);  
    readcount++;  
    if (readcount==1)  
        wait(wrt);  
    signal(mutex);  
    // reading is performed  
    wait(mutex);  
    readcount--;  
    if (readcount==0)  
        signal(wrt);  
    signal(mutex);  
} while (TRUE);
```

图 3.17 读者进程的结构

读者—写者问题及其解答可以进行推广，用来对某些系统提供读写锁。在获取读写锁时，需要指定锁的模式：读访问或写访问。当一个进程只希望读共享数据时，可申请读模式的读写锁；当一个进程希望修改数据时，则必须申请写模式的读写锁。多个进程可允许并发获取读模式的读写锁；而只有一个进程可为写操作而获取读写锁。

读写锁在以下情况下最为有用：

- 当可以区分哪些进程只需要读共享数据而哪些进程需要写共享数据；
- 当读者进程数比写进程多时。这是因为读写锁的建立开销通常比信号量或互斥锁要大，而这一开销可以通过允许多个读者来增加并发度的方法进行弥补。

3.5.3 哲学家进餐问题 (Dining-philosophers problem)

假设有 5 个哲学家，他们用一生来思考和吃饭。这些哲学家共用一个圆桌，每位都有一把椅子。在桌子中央是一碗米饭，在桌子上放着 5 只筷子（见图 3.18）。当某位哲学家思考时，他独自静静地思考，不和其他哲学家交互。时而，他会感到饥饿，并试图拿起与他相邻的两只筷子（筷子在他和他的左、右邻座之间）。一个哲学家一次只能拿起一只筷子。当一个饥饿的哲学家同时拿到两只筷子时，他就能吃了。吃完后，他会放下两只筷子，并重新进入思考。显然，一位哲学家不能从其他正在吃饭的哲学家手里抢走筷子。

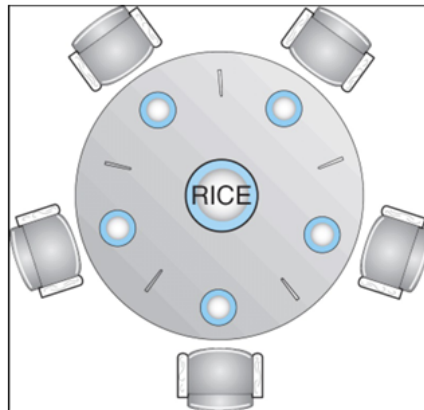


图 3.18 哲学家进餐问题的场景

哲学家进餐问题是一个典型的同步问题，这不是因为这一假想的问题本身有什么实际重要性，而是因为它是一个并发控制问题的典型例子。它需要在多个进程之间分配多个资源且不会出现死锁和饥饿的解决方案。

一种简单的解决方法是每只筷子都用一个信号量来表示。一个哲学家通过执行 `wait()` 操作试图获取相应的筷子，他会通过执行 `signal()` 操作以释放相应的筷子。因此，5 位哲学家的共享数据是：

```
semaphore chopstick[5];
```

其中所有 `chopstick` 的元素初始化为 1。哲学家 `i` 的进程结构如图 3.19 所示。

```
do {  
    wait(chopstick[i]);  
    wait(chopstick[(i+1)%5]);  
    // eat  
    signal(chopstick[i]);  
    signal(chopstick[(i+1)%5]);  
    // think  
} while (TRUE);
```

图 3.19 哲学家 `i` 的进程结构

虽然这一解答确保没有两个哲学家同时使用同一只筷子（即确保了互斥要求），但是这一解决仍

的筷子时，所有筷子的信号量均变为 0；然后，当每个哲学家试图拿右边的筷子时，他就会永远等待。这样，所有的哲学家最终都会饿死。

下面列出了几个可能解决死锁问题的方法：

- 最多只允许 4 个哲学家同时坐到桌子上。
- 只有两只筷子都可用时才允许一个哲学家拿起它们（他必须在临界区内拿起两只筷子）。
- 使用非对称解决方法，即奇数哲学家先拿起左边的筷子，接着拿起右边的筷子，而偶数哲学家先拿起右边的筷子，接着拿起左边的筷子。

最后，有关哲学家进餐问题的任何满意的解决必须确保没有一个哲学家会饿死。因为，即便找到了一个没有死锁的解决方案，并不能保证该方案消除了饿死的可能。

死锁:

1. 采用按序分配资源的策略可以预防死锁, 这是利用了哪个条件不成立?

A. 互斥 B. 循环等待 C. 不可抢占 D. 占有并等待

【答案】B

【解析】按序分配资源的策略将所有的资源按类型进行分类, 并赋予不同的序号。例如输入机的序号为 1, 打印机序号为 2, 磁盘机序号为 3 等。所有进程对资源的请求必须严格按资源序号递增的次序提出。这样在所形成的资源分配图中不可能再出现环路, 因而“循环等待”条件不成立。在采用这种策略时总有一个进程占据了较高序号的资源, 它继续请求的资源必然是空闲的, 因而进程可以一直向前推进。

2. 存在一进程等待序列 $\{P_1, P_2, \dots, P_n\}$, 其中 P_1 等待 P_2 所占有的某一资源, P_2 等待 P_3 所占有的资源, \dots 而 P_n 等待 P_1 所占有的资源形成一个_____。

A. 进程顺序推进 B. 进程循环等待环
C. 资源有序分配 D. 资源强占

【答案】B

【解析】循环等待: 若进程集合为 $\{P_0, P_1, \dots, P_n\}$, 那么存在这样的关系, P_0 等待的资源被 P_1 所占有, P_1 等待的资源被 P_2 所占有, P_{n-1} 等待的资源被 P_n 所占有, P_n 等待的资源被 P_0 所占有。例如, n 个进程 P_1, P_2, \dots, P_n , P_i ($i=1, \dots, n$) 因为申请不到资源 R_j ($j=1, \dots, m$) 而处于等待状态, 而 R_j 又被 P_{i+1} ($i=1, \dots, n-1$) 占有, P_n 欲申请的资源被 P_1 占有, 显然, 此时这 n 个进程的等待状态永远不能结束。

3. 互斥条件是指_____。

A. 某资源在一段时间内只能由一个进程占有, 不能同时被两个或两个以上的进程占有。
B. 一个进程在一段时间内只能占用一个资源。
C. 多个资源只能由一个进程占有。
D. 多个资源进程分配占有。

【答案】A

【解析】互斥条件: 一个资源一次只能被一个进程所使用。如果进程 A 正在使用资源; 如果另一进程 B 申请该资源, 那么进程 B 的申请得不到满足, 进程 B 只能等待, 直至该资源被释放。

4. 某计算机系统中有 8 台打印机, 由 K 个进程竞争使用, 每个进程最多需要 3 台打印机。该系统可能发生死锁的 K 的最小值是

A. 2 B. 3 C. 4 D. 5

【答案】C

【解析】假设 $K=3$ 个, 3 进程共享 8 台打印机, 每个进程最多可以请求 3 台打印机, 若 3 个进程都分别得到 2 台打印机, 系统还剩下 2 台打印机, 接下去无论哪个进程申请打印机, 都可以得到满足, 3 个进程都可以顺利执行完毕, 这种情况下不会产生死锁。假设 $K=4$, 4 个进程共享 8 台打印机, 若 4 个进程都分别得到 2 台打印机, 系统已经没有打印机了, 接下去无论哪个进程申请打印机, 都得不到满足, 产生了相互等待, 可能会发生死锁。因此答案是 C。

5. 设有一系统在某时刻的资源分配情况如下：

进程号	已分配资源	最大请求资源	剩余资源
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

请问：

- (1) 系统中各进程尚需资源数各是多少？
- (2) 当前系统安全吗？
- (3) 如果此时进程 P1 提出资源请求 (0, 4, 2, 0)，系统能分配给它吗？

参考答案：

- (1) 尚需资源数矩阵如下：

$$\text{Need} = \text{Max} - \text{Allocation}$$

Need				
	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

- (2) 系统是安全的，因为可以找到一个安全序列：<P0, P2, P3, P4, P1>

- (3) 如 P1 申请(0,4,2,0)，则：

$$\text{Request1}(0,4,2,0) \leq \text{need1}(0,7,5,0)$$

$$\text{Request1}(0,4,2,0) \leq \text{available}(1,5,2,0)$$

新的状态为

	Allocation	Max	Need	Available
P0	0 0 1 2	0 0 1 2	0 0 0 0	1 1 0 0
P1	1 4 2 0	1 7 5 0	0 3 3 0	
P2	1 3 5 4	2 3 5 6	1 0 0 2	
P3	0 6 3 2	0 6 5 2	0 0 2 0	
P4	0 0 1 4	0 6 5 6	0 6 4 2	

该状态是安全的，存在安全序列如<P0, P2, P3, P4, P1>，所以可以分配资源给 P1。

内存管理：

1. 把作业地址空间中使用的逻辑地址变成内存中物理地址称为_____。

- A. 加载
B. 地址绑定
C. 物理化
D. 逻辑化

【答案】B

【解析】内存中存储单元的地址，即加载到内存地址寄存器中的内存单元直接看到的地址通常称为物理地址；CPU生成的地址则通常称为逻辑地址，它是由用户的程序经过汇编或编译后形成目标代码，目标代码通常采用相对地址的形式。把相对地址转换成内存中的物理地址，这个过程称为地址绑定。

2. 在存储管理中，采用覆盖与交换技术的目的是_____。

- A. 节省主存空间
B. 物理上扩充主存容量
C. 提高CPU效率
D. 实现主存共享

【答案】A

【解析】覆盖是让同一内存区可以被不同的程序段重复使用，因此可以让那些不会同时执行的程序段共用同一个内存区；交换是指系统根据需要将内存中暂时不运行的作业全部移到外存，把外存中某些需要的作业移到内存。两者都是根据需要，提高内存利用率，节省内存使用空间。

3. 静态绑定在_____时进行；而动态绑定在_____时进行。

【答案】程序编译或装入内存 程序执行

【解析】将指令和数据地址绑定到内存地址可以在源程序处理流程的三个不同的阶段发生：编译时期、装入时期和执行时期。按照上面地址绑定的不同时期，可分为静态绑定和动态绑定。

4. 在页式管理中，页表的始址是存放在_____。

- A. 内存 B. 存储页面表中 C. 相联存储器中 D. 寄存器中

【答案】D

【解析】页表一般是存放在内存中的，即划分某些内存区域存放页表，而它的起始地址是存放在专门的寄存器中，以便地址转换机构能快速找到页表，这个寄存器称为页表始址寄存器。

5. 判断题：在分页存储管理中，减少页面大小，可以减少内存的浪费。所以页面越小越好。

()

【答案】错误

【解析】页面变小将导致页表的增大，即页表占用内存的增大。

6. 存储管理的主要研究内容是什么？

【分析】存储管理的主要研究内容是：

(1) 内存的分配和回收。记录内存空间的使用情况，实施内存的分配，回收系统或用户释放的内存空间。

- (2) 地址变换。即利用地址变换机构，将逻辑地址转换为物理地址。
- (3) 存储扩充。借助于虚拟存储技术，从逻辑上扩充内存，为用户提供比内存空间大的地址空间。
- (4) 存储保护。保证进入内存的各道作业都在自己的存储空间内运行，互不干扰。

7. 考虑一个由 8 个页面、每页 1024 个字节组成的逻辑空间，把它映射到容量为 32 个物理块的存储器中，试问逻辑地址和物理地址分别是多少位？为什么？

【答案】逻辑地址为 13 位，物理地址为 15 位。

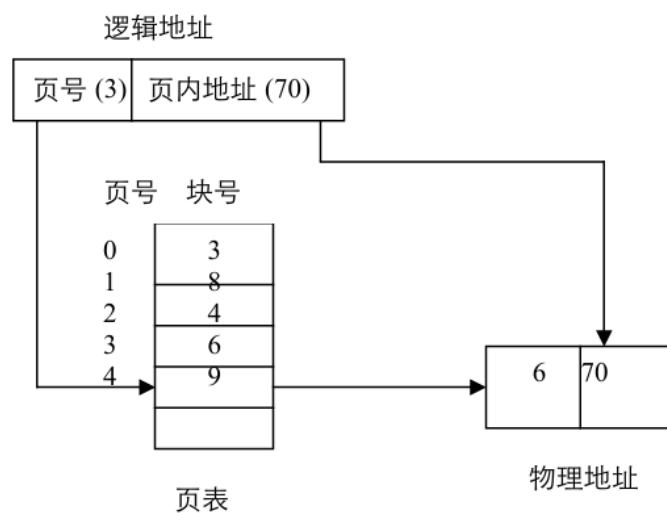
【分析】因为每页大小为 1024 字节，故页内地址需要 10 个二进制位描述：作业的逻辑地址空间有 8 页，页号需要 3 个二进制位。由此可知，逻辑地址共需 13 位。

因为每个物理块与页大小相同，即大小为 1024 字节，故块内地址需要 10 个二进制描述：内存空间容量为 32 块，块号需要 5 个二进制位。由此可知，物理地址共需要 15 位。

8. 假定某页式管理系统中，主存为 128KB，分成 32 块，块号为 0、1、2、3、…、31；某作业有 5 块，其页号为 0、1、2、3、4，被分别装入主存的 3、8、4、6、9 块中。有一逻辑地址为 [3, 70]。试求出相应的物理地址（其中方括号中的第一个元素为页号，第二个元素为页内地址，按十进制计算），并画图说明地址变换过程。

【答案】相应的物理地址为 24646。

【分析】块大小为 $128\text{KB}/32=4\text{KB}$ ，因为块与页面大小相等，所以每页为 4KB。第 3 页被装入到主存第 6 块中，故逻辑地址 [3, 70] 对应的物理地址为 $4\text{KB} \times 6 + 70 = 24576 + 70 = 24646$ 。其地址变换过程如下图。



页式系统的地址变换过程

9. 对一个将页表放在内存中的分页系统：

(2) 如果增加一个快表，且假定在快表中找到页表项的几率高达 90%，则有效访问时间又是多少（假定查找快表需花的时间为 0）？

【分析】每次访问数据时，若不使用快表，则需要两次访问内存，即先从内存的页表中读出页对应的块号，然后再根据形成的物理地址去存取数据；使用快表时，若能从快表中直接找到对应的页表项，则可立即形成物理地址去访问相应的数据，否则，仍需两次访问内存。

(1) 有效访问时间为： $2 \times 0.2 = 0.4\mu s$

(2) 有效访问时间为： $0.9 \times 0.2 + (1 - 0.9) \times 2 \times 0.2 = 0.22\mu s$

10. 一个基于动态分区存储管理的计算机的主存存储容量为 xxMB（初始为空），从低地址开始分配，分配和释放多次，分别采用首次适应（FirstFit）和最佳适应（BestFit）及最差(WorstFit)分配算法，画图并说明全部分配后主存中的空闲分区分别为多少。

虚拟内存：

1. 页式虚拟存储管理的主要特点是_____。

- A. 不要求将作业装入到主存的连续区域
- B. 不要求将作业同时全部装入到主存的连续区域
- C. 不要求进行缺页中断处理
- D. 不要求继续页面置换

【答案】B

【解析】页式存储管理的特点是不要求作业装入到内存的连续区域，而页式虚拟管理的特点是不要求作业同时全部装入到内存的连续区域。

2. 虚拟存储技术是_____。

- A. 扩充主存物理空间技术
- B. 扩充主存逻辑地址空间技术
- C. 扩充外存空间的技术
- D. 扩充输入/输出缓冲区技术

【答案】B

【解析】所谓虚拟存储器，是指仅把作业的一部分装入内存便可运行作业的存储器系统。具体地说，所谓虚拟存储器是指具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储器系统。实际上，用户所看到的大容量只是一种感觉，是虚的，故称之为虚拟存储器。虚拟存储技术是一种性能非常优越的存储器管理技术、故被广泛地应用于大、中、小型机器和微型机中。所以本题的答案是B。

3. 为使虚存系统有效地发挥其预期的作用，所运行的程序应具有的特性是_____。

- A. 该程序不应含有过多的 I/O 操作
- B. 该程序的大小不应超过实际的内存容量
- C. 该程序应具有较好的局部性 (Locality)
- D. 该程序的指令相关不应过多

【答案】C

【解析】程序具有局部性，即大约 20% 的部分程序占用了 80% 的运行时间，其余 80% 的部分则只占用大约 20% 的运行时间，正因为如此，可将暂时不需要的部分放在外存，要用时再调入主存。

3. 某虚拟存储器系统采用页式内存管理，使用 LRU 页面替换算法，考虑下面的页面访问地址流（每次访问在一个时间单位中完成）：

1 8 1 7 8 2 7 2 1 8 3 8 2 1 3 1 7 1 3 7

假定内存容量为 4 个页面，开始时是空的，则页面失效次数是_____。

- A. 4
- B. 5
- C. 6
- D. 7

【答案】B

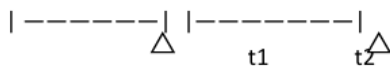
【解析】

页面走向	1	8	1	7	8	2	7	2	1	8	3	8	2	1	3	1	7	1	3	7
4 内存块	1	8 1	1 8	7 8	8 1	2 7	7 2	2 7	1 2	8 1	3 8	8 3	2 8	1 3	3 2	1 2	7 3	1 3	3 7	7 1
缺页	缺	缺		缺		缺											缺			

注意：题目可以出的很具体，例如访问的地址序列是：0300，0400，0500，0200 ……，现分配给该作业的主存共 2K 字节，页的大小为 512 字节。要知道实际意思就是 2KB/512B=4，内存窗口大小为 4，0300→3，0400→4，0500→5，0200→2，剩下求解同上。熟悉 FIFO、OPT(最优)和 LRU 这三个页面替换算法。

2. 在虚拟页式存储管理中，为解决颠簸问题，可采用工作集模型以决定分给进程的物理块数，有如下页面访问序列：

... .. 2 5 1 6 3 3 7 8 9 1 6 2 3 4 3 4 3 4 4 4 3 4 4 3



窗口尺寸 $\Delta = 9$ ，试求 t_1 、 t_2 时刻的工作集。

【分析】一个进程在时间 t 的工作集可形式化地定义为：

$W(t, h) = \{\text{在时间 } t-h \text{ 到 } t \text{ 之间所访问的一串页面}\}$

其中， h 为工作集窗口尺寸。

由题目所给条件可知， t_1 时刻的工作集为：{1, 2, 3, 6, 7, 8, 9}

t_2 时刻的工作集为：{3, 4}

文件系统:

1. 设置当前工作目录的主要目的是
- A. 节省外存空间 B. 节省内存空间
- C. 加快文件的检索速度 D. 加快文件的读/写速度

【答案】 C

【解析】文件系统最基本目标是实现按名存取。设置当前工作目录避免每一次搜索文件路径时都从根目录开始，这样可以加快文件的检索速度。

2. 从下面的描述中, 选择出一条错误的描述_____。
- A. 一个文件在同一系统中、不同的存储介质上的拷贝, 应采用同一种物理结构。
 - B. 文件的物理结构不仅与外存的分配方式相关, 还与存储介质的特性相关, 通常在磁带上只适合使用顺序的存储结构。
 - C. 采用顺序(连续)存储结构的文件既适合进行顺序访问, 也可以进行直接访问。
 - D. 虽然磁盘是随机访问的设备, 但其中的文件也可使用顺序存储结构。

【答案】 A

【解析】一个文件存放在磁带中通常采用连续存放，文件在硬盘上一般不采用连续存放方法，不同的文件系统存放的方法是不一样的，例如，硬盘的某个分区，其采用 Windows NTFS 文件系统，把这个分区中的文件刻录到光盘上，这个文件在硬盘和光盘上的存储方式是不一样的。

3. 某文件中共有 3 个记录。每个记录占用一个磁盘块，在一次读文件的操作中，为了读出最后一个记录，不得不读出了其他的 2 个记录。根据这个情况，可知这个文件所采用的结构是_____。
- A. 顺序结构 B. 链接结构 C. 索引结构 D. 顺序结构或链接结构

【答案】 B

【解析】链接文件的缺点是在随机存取某一个记录前需要化多次盘 I/O 操作读该记录前的文件信息以取得该记录的盘块号，才能存取该记录，所以链接文件只适用于顺序存取文件。

4. 设有一个包含 1000 个记录的索引文件，每个记录正好占用一个物理块。一个物理块可以存放 10 个索引表目。建立索引时，一个物理块应有一个索引表目。试问：
- (1) 该文件至少应该建立____级索引〔假定一级索引占用一个物理块〕？
- (2) 索引应占____个物理块？
- (1) A. 1 B. 2 **C. 3** D. 4
- (2) A. 1 B. 11 **C. 111** D. 1111

【答案】 (1)C (2)C

【解析】一个包含 1000 个记录的索引文件，每个记录正好占用一个物理块，文件本身应占 1000 个物理块。建立索引时，一个物理块应有一个索引表目，该文件共有 1000 个表目。一个物理块可以存放 10 个索引表目，该级索引需要 100 个物理块来存放 1000 个索引表目。由于该级索引表超过一个物理块，为此需为索引表再建索引，该级索引 100 个物理块需 10 个索引表目，100 个索引表目需 10 个物理块来存放。由于该级索引表仍超过一个物理块，还需为索引表再建索引，该级索引 10 个物理块需 10 个索引表目，10 个索引表目需 1 个物理块来存放。可知应建立三级索引，索引应占 $1+10+100=111$ 个物理块。

5. 下列文件的物理结构中, 不利于文件长度动态增长的文件物理组织形式是_____。
- A. 连续 B. 链接 C. 索引 D. 链接索引

【答案】 A