操作系统原理离线作业

一、单选题

1. 进程 P0 和 P1 的共享变量定义及其初值为

```
boolean flag[2];
int turn=0;
flag[0]=FALSE; flag[1]=FALSE;
```

若进程 P0 和 P1 访问临界资源的类 C 代码实现如下：

```
void P0() //P0进程
 { while (TURE) {
    flag[0]=TRUE; turn = 1;
    while (flag[1] && turn == 1) ;
     临界区;
    flag[0] = FALSE;
  }
}
void P1()   //P1进程
{ while (TURE) {
    flag[1]=TRUE; turn = 0;
    while (flag[0] && turn == 0) ;
     临界区;
    flag[1] = FALSE;
  }
}
```

则并发执行进程 P0 和 P1 时产生的情况是：D

A.不能保证进程互斥进入临界区、会出现“饥饿”现象　　B.不能保证进程互斥进入临界区、不会出现“饥饿”现象

C.能保证进程互斥进入临界区、会出现“饥饿”现象　　D. 能保证进程互斥进入临界区、不会出现“饥饿”现象

2.有两个进程 P1 和 P2 描述如下:

```
shared data:
int counter = 6;
P1 :
    Computing;
    counter=counter+1;
P2 :
    Printing;
    counter=counter-2;
```

两个进程并发执行，运行完成后，counter 的值不可能为 C 。

A. 4　　B. 5　　C. 6　　D. 7

3.某计算机采用二级页表的分页存储管理方式，按字节编址，页大小为 2^{10} 字节，页表项大小为 2 字节，逻辑地址结构为：

页目录号 页号 页内偏移量

逻辑地址空间大小为 2^{16} 页，则表示整个逻辑地址空间的页目录表中包含表项的个数至少是 B

A. 64　　B. 128　　C. 256　　D. 512

4.在动态分区系统中，有如下空闲块:

空闲块	块大小（KB）	块的基址
1	80	60
2	75	150
3	55	250
4	90	350

此时,某进程 P 请求 50KB 内存,系统从第 1 个空闲块开始查找，结果把第 4 个空闲块分配给了 P 进程 ，请问是用哪一种分区分配算法实现这一方案？ C

A. 首次适应　　B. 最佳适应　　C. 最差适应　　D. 下次适应

5.在一页式存储管理系统中，页表内容如下所示。

页号	帧号
0	2
1	1
2	8

若页大小为 1K，逻辑地址的页号为 2，页内地址为 451，转换成的物理地址为　A

A. 8643　　B. 8192　　C. 2048　　D. 2499

6.采用段式存储管理的系统中，若地址用 32 位表示，其中 20 位表示段号，则允许每段的最大长度是　B

A. 2^{24}　　B. 2^{12}　　C. 2^{10}　　D. 2^{32}

7.在一段式存储管理系统中，某段表的内容如下:

段号	段首址	段长
0	100K	35K
1	560K	20K
2	260K	15K
3	670K	32K

若逻辑地址为(2, 158)，则它对应的物理地址为__B___。

A. 100K+158　　B. 260K+158　　C. 560K+158　　D. 670K+158

8.一个分段存储管理系统中，地址长度为 32 位，其中段长占 8 位，则最大段长是 C

A. 2^8字节　　B. 2^{16}字节　　C. 2^{24}字节　　D. 2^{32}字节

9.有一请求分页式存储管理系统，页面大小为每页 100 字节，有一个 50×50 的整型数组按行为主序连续存放，每个整数占两个字节，将数组初始化为 0 的程序描述如下：

```
int A[50][50];
for (int i = 0; i < 50; i++)
  for (int j = 0; j < 50; j++)
    A[i,j] = 0;
```

若在程执行时内存只有一个存储块用来存放数组信息，试问该程序执行时产生__B__次缺页中断。

A. 1　　B. 50　　C. 100　　D. 2500

10.一台计算机有 4 个页框，装入时间、上次引用时间、和每个页的访问位 R 和修改位 M，如下所示：

页	装入时间	上次引用时间	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

采用 FIFO 算法将淘汰__C__页；　　A. 0　　B. 1　　C. 2　　D. 3

11.一台计算机有 4 个页框，装入时间、上次引用时间、和每个页的访问位 R 和修改位 M，如下所示：

页	装入时间	上次引用时间	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

采用 NRU 算法将淘汰__A__页；　　A. 0　　B. 1　　C. 2　　D. 3

12.一台计算机有 4 个页框，装入时间、上次引用时间、和每个页的访问位 R 和修改位 M，如下所示：

页	装入时间	上次引用时间	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

采用 LRU 算法将淘汰__B__页；　　A. 0　　B. 1　　C. 2　　D. 3

13.一台计算机有 4 个页框，装入时间、上次引用时间、和每个页的访问位 R 和修改位 M，如下所示：

页	装入时间	上次引用时间	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

采用第二次机会算法将淘汰___A___页；　　A. 0　　B. 1　　C. 2　　D. 3

二、综合题

1.4 在所列的两种设置中，哪些功能需要操作系统提供支持？ (a)手持设备(b)实时系统。

a. 批处理程序 b. 虚拟存储器 c. 分时

答：对于实时系统来说，操作系统需要以一种公平的方式支持虚拟存储器和分时系统。对于手持系统，操作系统需要提供虚拟存储器，但是不需要提供分时系统。批处理程序在两种环境中都是非必需的。

1.17 列出下列操作系统的基本特点：

a.批处理 b.交互式 c.分时 d.实时 e.网络 f.并行式 g.分布式 h.集群式 i.手持式

答： a.批处理：具有相似需求的作业被成批的集合起来，并把它们作为一个整体通过一个操作员或自动作业程序装置运行通过计算机。通过缓冲区，线下操作，后台和多道程序，运用尝试保持 CPU 和 I/O 一直繁忙，从而使得性能被提高。批处理系统对于运行那些需要较少互动的大型作业十分适用。它们可以被更迟地提交或获得。

a.交互式：这种系统由许多短期交易构成，并且下一个交易的结果是无法预知的。从用户提交到等待结果的响应时间应该是比较短的，通常为 1 秒左右。

b.分时：这种系统使用 CPU 调度和多道程序来经济的提供一个系统的人机通信功能。CPU 从一个用户快速切换到另一个用户。以每个程序从终端机中读取它的下一个控制卡，并且把输出的信息正确快速的输出到显示器上来替代用 soopled card images 定义的作业。

c.实时：经常用于专门的用途。这个系统从感应器上读取数据,而且必须在严格的时间内做出响应以保证正确的性能。

d.网络：提供给操作系统一个特征，使得其进入网络，比如;文件共享。

e.并行式：每一个处理器都运行同一个操作系统的拷贝。这些拷贝通过系统总线进行通信。

f.分布式：这种系统在几个物理处理器中分布式计算，处理器不共享内存或时钟。每个处理器都有它各自的本地存储器。它们通过各种通信线路在进行通信，比如：一条高速的总线或一个本地的网络。

g.集群式：集群系统是由多个计算机耦合成单一系统并分布于整个集群来完成计算任务。

i.手持式：一种可以完成像记事本，email 和网页浏览等简单任务的小型计算机系统。手持系统与传统的台式机的区别是更小的内存和屏幕以及更慢的处理能力。

2.3 讨论向操作系统传递参数的三个主要的方法。

答：1.通过寄存器来传递参数　　2.寄存器传递参数块的首地址　　3.参数通过程序存放或压进堆栈中，并通过操作系统弹出堆栈。

2.12 采用微内核方法来设计系统的主要优点是什么？在微内核中如何使客户程序和系统服务相互作用？微内核方法的缺点是什么？

答：a）增加一个新的服务不需要修改内核　　b）在用户模式中比在内核模式中更安全、更易操作

c）一个简单的内核设计和功能一般导致一个更可靠的操作系统

用户程序和系统服务通过使用进程件的通信机制在微内核中相互作用，例如发送消息。这些消息由操作系统运送。微内核最主要的缺点是与进程间通信的过度联系和为了保证用户程序和系统服务相互作用而频繁使用操作系统的消息传递功能。

3.2 问:描述一下内核在两个进程间进行上下文功换的动作.

答：总的来说，操作系统必须保存正在运行的进程的状态，恢复进程的状态。保存进程的状态主要包括 CPU 寄存器的值以及内存分配，上下文切换还必须执行一些确切体系结构的操作，包括刷新数据和指令缓存。

进程关联是由进程的 PCB 来表示的，它包括 CPU 寄存器的值和内存管理信息等。当发生上下文切换时，内核会将旧进程的关联状态保存在其 PCB 中，然后装入经调度要执行的新进程的已保存的关联状态。

3.4 如下所示的程序，说明 LINE A 可能会输出什么？

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int value=8;
int main()
{
pid_t pid;
```

```
    /* fork a child process */
    pid = fork();
if (pid == 0) { /* child process */
         value +=15;
    }
    else { /* parent process */
         /* parent will wait for the child to complete */
         wait(NULL);
         printf(" Parent :value= %d\n",value);/*LINE A*/
         exit(0);
    }
}
```

答：Parent :value=8。

4.4 在多线程程序中，以下哪些程序状态组成是被线程共享的？ a.寄存值 b.堆内存 c.全局变量 d.栈内存

答：一个线程程序的线程共享堆内存和全局变量，但每个线程都有属于自己的一组寄存值和栈内存。

4.7 由图 4.11 给出的程序使用了 Pthread 的应用程序编程接口（API），在程序的第 c 行和第 p 行分别会输出什么？

```
#include <pthread.h>
#include <stdio.h>
int value=0;
void  *runner(void *param); /* the thread */
int main(int argc, char *argv[])
{
int pid;
pthread_t tid;
pthread_attr_t attr;
    pid = fork();
    if (pid == 0)  {/* child process */
      pthread_attr_init(&attr);
      pthread_create(&tid, &attr, runner, NULL);
      pthread_join(tid, NULL);
      printf(“CHILD: value = %d”, value); /* LINE C*/
    }
    else if (pid > 0) {/* parent process */
          wait(NULL);
          printf(“PARENT: value = %d”, value); /* LINE P */
    }
}
void *runner(void *param) {
     value=10;
     pthread_exit(0);
}
```

答：c行会输出10，p行会输出0.

5.4 考虑下列进程集，进程占用的 CPU 区间长度以毫秒来计算：

进程	区间时间	优先级
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

假设在时刻 0 以进程 P_1，P_2，P_3，P_4，P_5的顺序到达。

a.画出 4 个 Gantt 图分别演示用 FCFS、SJF、非抢占优先级（数字小代表优先级高）和 RR（时间片＝1）算法调度时进程的执行过程。b.每个进程在每种调度算法下的周转时间是多少？

c.每个进程在每种调度算法下的等待时间是多少？d.哪一种调度算法的平均等待时间对所有进程而言最小？

答：

a.甘特图

FCFS

P1										P2	P3		P4	P5					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

SJF

P2	P4	P3		P5					P1										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

Non-preemptive Priority

P2	P5					P1										P3		P4	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

RR(quantum=1)

P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1	P5	P1	P5	P1	P1	P1	P1	P1	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

b. Turnaround Time

Process	FCFS	SJF	NPP	RR(quantum=1)
P1	10	19	16	19
P2	11	1	1	2
P3	13	4	18	7
P4	14	2	19	4
P5	19	9	6	14
Average	13.4	7.2	12	9.2

c. Waiting Time

Process	FCFS	SJF	NPP	RR(quantum=1)
P1	0	9	6	9
P2	10	0	0	1
P3	11	2	16	5
P4	13	1	18	3
P5	14	4	1	9
Average	9.6	3.2	8.2	5.4

d.SJF

5.5 下面哪些算法会引起饥饿　　a.先来先服务　　b.最短作业优先调度　　c.轮转法调度　　d.优先级调度

答：最短作业优先调度和优先级调度算法会引起饥饿。

5.7 考虑一个运行 10 个 I/O 约束（型）任务和一个 CPU 约束（型）任务的系统。假设，I/O 约束任务每进行 1 毫秒的 CPU 计算发射一次 I/O 操作，但每个 I/O 操作的完成需要 10 毫秒。同时，假设上下文切换要 0.1 毫秒，所有的进程都是长进程。对一个 RR 调度来说，以下情况时 CPU 的利用率是多少：　　a.时间片是 1 毫秒　　b.时间片是 10 毫秒

答：a.时间片是 1 毫秒：不论是哪个进程被调度，这个调度都会为每一次的上下文切换花费一个 0.1 毫秒的上下文切换。CPU 的利用率是 1/1.1*100=92%。

b.时间片是 10 毫秒：这 I/O 限制任务会在使用完 1 毫秒时间片后进行一次上下文切换。这个时间片要求在所有的进程间都走一遍，因此，10*1.1+10.1(因为每个 I / O 限定任务执行为 1 毫秒，然后承担上下文切换的任务，而 CPU 限制任务的执行 10 毫秒在承担一个上下文切换之前）。因此，CPU 的利用率是 20、21.1*100=94%。

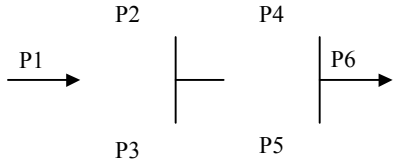
6.01 在生产者和消费者问题中，信号量 mutex，empty，full 的作用是什么？如果对调生产者进程中的两个 wait 操作和两个 signal 操作，则可能发生什么情况？

答：信号量 mutex 的作用是保证各生产者进程和消费者进程对缓冲池的互斥访问。信号量 empty 和 full 均是资源信号量，它们分别对应于缓冲池中的空闲缓冲区和缓冲池中的产品，生产者需要通过 wait(empty)来申请使用空闲缓冲区，而消费者需要通过 wait(full)才能取得缓冲中的产品，可见，这两个信号量起着同步生产者和消费者的作用，它们保证生产者不会将产品存放到满缓冲区中，而消费者不会从空缓冲区中取产品。

在生产者—消费者问题中，如果将两个 wait 操作，即 wait(full)和 wait(mutex)互换位置，或者 wait(empty)和 wait(mutex)互换位置，都可能引起死锁。考虑系统中缓冲区全满时，若一生产者进程先执行了 wait(mutex)操作并获得成功，当再执行 wait(empty)操作时，它将因失败而进入阻塞状态，它期待消费者执行 signal(empty)来唤醒自己，在此之前，它不可能执行 signal(mutex)操作，从而使企图通过 wait(mutex)进入自己的临界区的其他生产者和所有的消费者进程全部进入阻塞状态，系统进入死锁状态。类似地，消费者进程若先执行 wait(mutex)，后执行 wait(full)同样可能造成死锁。

signal(full)和 signal(mutex)互换位置，或者 signal(empty)和 signal(mutex)互换位置，则不会引起死锁，其影响只是使某个临界资源的释放略为推迟一些。

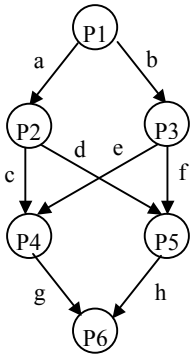
6.02 一组合作进程，执行顺序如下图。请用 wait、signal 操作实现进程间的同步操作。



各进程的执行顺序

答：如图示并发进程之间的前趋关系，为了使上述进程同步，可设置 8 个信号量 a、b、c、d、e、f、g、h，它们的初值均为 0，而相应的进程可描述为（其中“…”表示进程原来的代码）：

```
main( )
cobegin{
  P1( ) {  …;  signal(a);  signal(b);  }
  P2( ){  wait(a);  …;  signal(c);  signal(d);  }
  P3( ){  wait(b);  …;  signal(e);  signal(f);  }
  P4( ){  wait(c);  wait(e);  …;  signal(g);  }
  P5( ){  wait(d);  wait(f);  …;  signal(h);  }
  P6( ){  wait(g);  wait(h);  …;  }
}coend
```



6.03 在生产者和消费者问题中，多个生产者进程（Producer Process）和多个消费者进程（Consumer Process）共享一个大小为 8 的缓冲区，他们的信号量和共享变量设置如下：

```
int nextc=0, nextp=0, buf[8];
semaphore full; empty; mutex;
```

生产者进程和消费者进程问题的算法描述如下：

```
Producer Process:                    Consumer Process:
int itemp;                                  int itemc;
while(1){                                   while(1){
```

```
1    itemp = rand(); // Generate a number
2    wait(empty);
3    wait(mutex);
4    buf[nextp]=itemp;
5    nextp=(nextp+1)%8;
6    signal(mutex);
7    signal(full);
  }
```

```
1    wait(full);
2    wait(mutex);
3    itemc=buf[nextc];
4    nextc=(nextc+1)%8;
5  signal(mutex);
6    signal(empty);
7    cout << itemc << endl;
     }
```

（1）生产者进程和消费者进程的临界区是哪些？

（2）信号量full、empty和mutex的初值是多少？

（3）如果对调生产者进程中的两个P操作即第2行和第3行，以及对调消费者进程中的两个P操作即第1行和第2行，如下所示。可能发生什么情况？

```
Producer Process                          Consumer Process
…                                         …
1 itemp = rand(); // Generate a number    1 wait(mutex);
2 wait(mutex);                            2 wait(full);
3 wait(empty);                            3 itemc=buf[nextc];
…                                         …
```

（4）上面的生产者和消费者同步算法有一个缺点，在有空缓冲区时，当消费者进程正在临界区时，生产者进程必须等待，反之亦然。您如何可以解决这个问题，以提高生产者和消费者进程之间并发？写出新的生产者进程和消费者进程的同步算法。

答：（1）生产者进程的临界区是第4行和第5行；消费者进程的临界区是第3行和第4行。

（2）信号量full、empty和mutex的初值分别是：empty = 8 ， full = 0 ， mutex = 1 。

（3）系统可能会产生死锁。例如，生产者进程得到信号量mutex，但是没有空缓冲区即empty≤0时，此时生产者进程阻塞；而消费者进程又无法得到信号量mutex，此时消费者进程也阻塞，系统产生了死锁。

（4）增加一个信号量mutex1，初值为1，其算法如下：

```
Producer Process
int itemp;
while(1){
1    itemp = rand(); // Generate a number
2    wait(empty);
3    wait(mutex1);
4    buf[nextp]=itemp;
5    nextp=(nextp+1)%8;
6    signal(mutex1);
7    signal(full);
   }
```

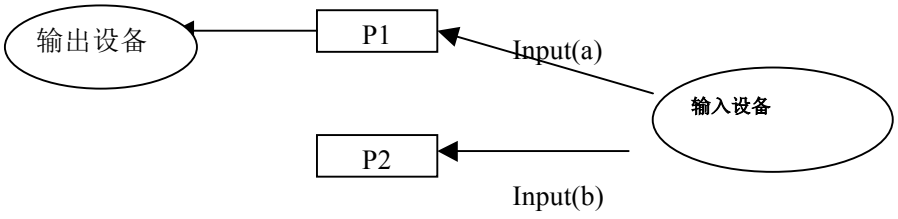
```
Consumer Process
    int itemc;
while(1){
1     wait(full);
2     wait(mutex);
3     itemc=buf[nextc];
4   nextc=(nextc+1)%8;
5   signal(mutex);
6     signal(empty);
7     cout << itemc << endl;
      }
```

6.04 有 2 个合作的进程 P1、P2 。他们从一台输入设备读入数据，P1 进程读入数据 a，P2 进程读入数据 b。输入设备是一台独享设备 。两个进程做如下计算：

P1: x = a + b

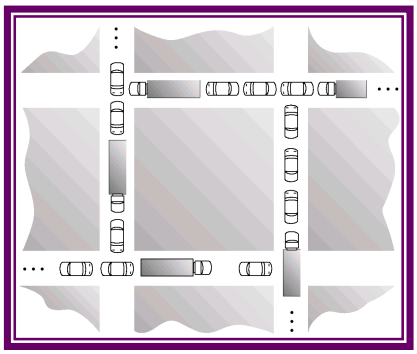
P2: y = a * b

计算完成后结果的x、y由进程P1输出。用信号量实现P1、P2同步算法。



答：设置三个信号：s1表示数据a是否读入，s2表示数据b是否读入，s3表示数据y=a*b计算是否完成。P1和P2两个进程的同步算法如下：

```
semaphore s1=0, s2=0, s3=0;
main()
cobegin{
P1:                                P2:
{                                  {
     input (a);                         wait(s1);
     signal(s1);                        input (b);
     wait(s2);                          signal(s2);
     x=a+b;                        y=a*b;
     wait(s3) ;                         signal(s3);
     Print (x,y,z);
}                                  }
}coend
```



7.1 假设有如下图所示的交通死锁情况：

(1) 说明产生死锁的 4 个必要条件在此处成立。

(2) 给出一个避免死锁的简单规则。

答：（1）在此处，产生死锁的四个必要条件如下：

1) 互斥条件。每个车道的每段道路只能被一辆车占用。

2) 请求与保持条件。每个车队占用了一个车道，并请求前方的车道，即使需等待前方车道上的车队驶离，它仍将持有已占用的车道。

3) 不抢占（剥夺）条件。在前方的车道被其它车队占用时，因为是单车道，而其它车队又不会后退，所以无法从其它车队处抢占车道。

4) 环路等待条件。向东行驶的车队等待向北行驶的车队让出车道，向北行驶的车队等待向西行驶的车队让出车道，向西行驶的车队等待向南行驶的车队让出车道，而向南行驶的车队则等待向东行驶的车队让出车道。故存在一循环等待链。

（2）增加一个约束条件：只有前方两个路口都空闲时，才能占用第一个路口。或者，可在十字路口设置一交通信号灯，并使南北方向的两个车队和东西方向的两个车队互斥地使用十字路口，便可避免交通死锁。

7.11 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 A B C D	最大请求资源 A B C D	剩余资源 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？

答：（1）尚需资源数矩阵如下：

Need = Max - 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：

Request1(0,4,2,0) <=need1(0,7,5,0)

Request1(0,4,2,0) <= 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。

8.3 某系统有五个固定分区，其长度依次为 100K, 500K, 200K, 300K, 600K。今有四个进程，对内存的需求分别是 212K, 417K, 112K, 426K。当分别用 First-fit, Best-fit, Worst-fit 算法响应这四个进程的内存申请时，请分别给出系统的内存分配动态。哪种算法最有效？

答：根据First-fit、Best-fit、Worst-fit算法，计算结果如下：

First-fit:

212K进程装到500K分区

417K进程装到600K分区

112K进程装到200K分区

426K进程暂时等待

Best-fit:

212K进程装到300K分区

417K进程装到500K分区

112K进程装到200K分区

426K进程装到600K分区

Worst-fit:

212K进程装到600K分区

417K进程装到500K分区

112K进程装到300K分区

426K进程暂时等待

8.5 对下列问题，试比较连续内存分配方案、纯段式分配方案、纯页式分配方案中的内存组织方法：

a. 外部碎片　　b. 内部碎片　　c. 共享跨进程代码的能力

答：连续内存分配会产生外部碎片，因为地址空间是被连续分配的，当旧进程结束，新进程初始化的时候，洞会扩大。连续内存分配也不允许进程共享代码，因为一个进程的虚拟内存段是不被允许闯入不连续的段的。纯段式分配也会产生外部碎片，因为在物理内存中，一个进程的段是被连续放置的，以及当死进程的段被新进程的段所替代时，碎片也将会产生。然而，段式分配可以使进程共享代码；比如，两个不同的进程可以共享一个代码段，但是有不同的数据段。纯页式分配不会产生外部碎片，但会产生内部碎片。进程可以在页 granularity 中被分配，以及如果一页没有被完全利用，它就会产生内部碎片并且会产生一个相当的空间浪费。在页 granularity，页式分配也允许进程共享代码。

8.9 考虑一个分页式存储管理系统，其页表常驻内存。

（1）如果内存访问耗时200 ns，那么，访问内存中的数据需要多长时间？

（2）如果引入联想寄存器，而且75%的页面可以从关联寄存器中找到，那么，此时的有效访问时间为多少？(假设访问关联寄存器的时间可以忽略)

答：（1）400纳秒，其中，200纳秒访问页表，200纳秒访问内存中的数据。

（2）有效访问时间 = 0.75 *（200纳秒访问内存数据+0纳秒访问关联寄存器）+ 0.25 *（200纳秒访问内存数据+200纳秒访问页表）= 250纳秒

8.12 假设有下列段表：

段	基地址	段长度
0	219	600
1	2300	14

2 90 100

3 1327 580

4 1952 96

下列逻辑地址对应的物理地址是什么？

（1） 0,430 （2）1,10 （3）2,500 （4）3,400 （5）4,112

答：（1）219 + 430 = 649 （2）2300 + 10 = 2310

（3） 第2段的有效长度是100。段内偏移量500超过了这个上限，所以这是个非法地址

（4）1327 00 = 1727 （5）第4段的有效长度是96。段内偏移量112超过了这个上限，所以这是个非法地址

9.5 假设一个“按需调页”虚拟存储空间，页表由寄存器保存。在存在空闲页帧的条件下，处理一次缺页的时间是 8 毫秒。如果没有空闲页面，但待换出页面并未更改，处理一次缺页的时间也是 8 毫秒。如果待换出页面已被更改，则需要 20 毫秒。访问一次内存的时间是 100 纳秒。假设 70%的待换出页面已被更改，请问缺页率不超过多少，才能保证有效访问时间小于或等于 200 纳秒？

答：设缺页率为P。题目并没有明确，当缺页中断时，内存中是否有空闲页帧，所以假设内存总是忙的。

访问内存中页面：(1 - P) * 100ns

页面不在内存，但不需要保存待换出页面：P * (1 - 70%) * (8ms+100ns)

页面不在内存，但需要保存待换出页面：P * 70% * (20ms+100ns)

所以，有效访问时间=(1 - P) * 100ns + P * (1 - 70%) * (8ms+100ns) + P * 70% * (20ms+100ns) = 200ns

P = 0.000006

9.10 对一个请求调页系统测得如下数据：

- CPU利用率20%
- 用作页面交换的磁盘的利用率97.7%
- 其它I/O设备利用率5%

下列措施中，哪些会改善CPU利用率（如果有的话），请说明理由：

(1) 安装一个更快的CPU

(2) 安装一个更大容量的磁盘用作页面交换

(3) 增加并发进程数

(4) 减少并发进程数

(5) 安装更多内存

(6) 安装更快的硬盘，或安装更多的硬盘和控制器

(7) 增加一个预取页面算法

(8) 增加页面长度

答：a.首先判断系统正在频繁地进行换页操作。所以，减少并发进程数会显著地减少换页操作，提高CPU的利用率。其它措施也有些效果，例如，安装更多内存。

b.安装一个更快的CPU。没用。

c.安装一个更大容量的磁盘用作页面交换。没用，交换空间本来就足够了。

d.增加并发进程数。没用，情况将会更糟。

e.减少并发进程数。效果明显。

f.安装更多内存。可能会有效果，因为空闲页帧增加了，换页的几率将相对减少。

g.安装更快的硬盘，或安装更多的硬盘和控制器。效果不明显。

h.增加一个预取页面算法。效果不确定。

i.增加页面长度。如果顺序访问居多，则会减少缺页次数。如果随机访问居多，因为单

个页面占用更大的物理空间，页帧总数减少，所以缺页次数会增加；因为页面长度增加，页面的传输时间会增加。综上，此方案的效果不确定。

9.14 一页式虚拟存储系统，用于页面交换的磁盘的平均访问、传输时间是 20 毫秒。页表保存在主存，访问时间 1 微秒。也就是说，每引用一次指令或数据，需要访问两次内存。为改善性能，我们可以增设一个关联寄存器。如果页表项在关联寄存器里，则只要访问一次内存就够了。假设 80%的访问，其页表项在关联寄存器中；剩下的 20%里，10%的访问（即总数的 2%）会产生缺页。请计算有效访问时间。

答：有效访问时间 = 80% * 1微秒 + (1-80%)((1-10%) * 1微秒 * 2 + 10% * (1微秒 * 2 + 20毫秒))= 0.8+0.2 * (0.9 * 2+0.1*20002)

= 0.8+0.2 * 2002

= 401.2微秒

9.01 在某请求分页管理系统中，一个作业共 5 页，作业执行时依次访问如下页面：1，4，3，1，2，5，1，4，2，1，4，5，若分配给该作业的主存块数为 3，分别采用 FIFO、LRU，试求出缺页中断的次数及缺页率。(要求画出页面置换情况表)

答：(1)采用 FIFO 页面置换算法，其缺页情况如表所示：

FIFO 页面置换算法的缺页情况

页面走向	1	4	3	1	2	5	1	4	2	1	4	5
块 1	1	1	1		2	2	2	4	4			4
块 2		4	4		4	5	5	5	2			2
块 3			3		3	3	1	1	1			5
缺页	√	√	√		√	√	√	√	√			√

缺页中断次数为 9，缺页率为 9/12=75%。

(2)采用 LRU 页面置换算法，其缺页情况如表所示。

LRU 页面置换算法的缺页情况

页面走向	1	4	3	1	2	5	1	4	2	1	4	5
块 1	1	1	1		1	1		1	1			1
块 2		4	4		2	2		4	4			4
块 3			3		3	5		5	2			5
缺页	√	√	√		√	√		√	√			√

缺页中断次数为 8，缺页率为 8/12=67%。

10.1 假设有一个文件系统，它里面的文件被删除后，当连接到该文件的链接依然存在时，文件的磁盘空间会再度被利用。如果一个新的文件被创建在同一个存储区域或具有同样的绝对路径名，这会产生什么问题？如何才能避免这些问题？

答：假设F1是旧的文件，F2是新的文件。当一个用户通过已存在的链接访问F1，实际却是访问F2。这里使用的是对文件F1的存取保护而不是与文件F2相关的存储保护。

采用删除指向一个已删除文件的所有链接的方法避免该问题。可以通过几种方法实现：

1．设置一个记录指向一个文件的所有链接的链表，当这个文件被删除时，删掉这些链接。

2．保留这些链接，当试图访问一个已被删除的文件时，删掉这些链接。

3．设置一个文件的指针链表（或计数器），当指向该文件的所有指针被删除时才真正删除这个文件。

10.9 有些系统文件提供文件共享时候只保留文件的一个拷贝，而另外的一个系统则是保留多个拷贝，对共享文件的每一个用户提供一个拷贝，论述这种方法的相对优点。

答：在一个单一的复制，同时更新了一个文件可能会导致用户获得不正确的信息，文件被留在了不正确的状态. 随着多份拷贝，它会浪费存储而且各种副本可能不一致

11.6 假设一个在磁盘上的文件系统，其中逻辑块和物理块大小为 512 字节。假定每个文件的信息已经在内存中，对于三种分配策略中的每一种（连续、链接、索引），请回答下面这些问题。

（1）说明在这个系统中是如何实现从逻辑地址到物理地址映射的？（对于索引分配，假设文件的长度总是小于 512 块）。

（2）如果当前位于逻辑块10（即最后一次访问的逻辑块是10），且希望访问逻辑块4，必须从磁盘上读多少个物理块？

答：令 Z 是开始逻辑地址（块号）。

a. 若使用连续分配策略时。用 512 去除逻辑地址，则 X 和 Y 分别表示得到的整数和余数。

（1）将 X 加上 Z 得到物理块号，Y 为块内的位移　　（2）1

b. 若使用链接分配策略。用 511 去除逻辑地址，则 X 和 Y 分别表示得到的整数和余数。

（1）查找链表到第 X+1 块，Y+1 位该块内的位移量　　（2）4

c. 若使用索引分配策略。用 512 去除逻辑地址，则 X 和 Y 分别表示得到的整数和余数。

（1）把索引块读入内存中，则物理块地址存放在索引块在第 X 位置中，Y 为块内的位移量　　（2）2

11.01 考虑一个含有 100 块的文件。假如文件控制块（和索引块,当用索引分配时）已经在内存中。当使用连续、链接、单级索引分配策略时，各需要多少次磁盘 I/O 操作？假设在连续分配时，在开始部分没有扩张的空间，但在结尾部分有扩张空间，并且假设被增加块的信息已在内存中：

（1）在开始增加一块。

（2）在中间增加一块。

（3）在末端增加一块。

（4）在开始删除一块。

（5）在中间删除一块。

（6）在末端删除一块。

答：各种策略相应的磁盘 I/O 操作次数如表

	连续	链接	索引
a.	201	1	1
b.	101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

11.02 有一磁盘组共有 10 个盘面，每个盘面上有 100 个磁道，每个磁道有 16 个扇区。假设分配以扇区为单位。

（1） 若使用位示图管理磁盘空间，问位示图需要占用多少空间？

（2） 若空白文件目录的每个表目占用 5 个字节，问什么时候空白文件目录大于位示图？

答：空白文件目录是管理磁盘空间的一种方法，该方法将文件存储设备上的每个连续空闲区看作一个空白文件，系统为所有空白文件单独建立一个目录，每个空白文件在这个目录中占一个表项；表项的内容至少包括第一个空白块的地址（物理块号）、空白块的数目。

（1）由题设所给条件可知，磁盘组扇区总数为 16*100* 10=16000（个）

因此，使用位示图描述扇区状态需要的位数为 16000（位）/8(位/字节）=2000（字节）

(2)已知空白文件目录的每个表项占 5 个字节．而位示图需占 2000 字节．即 2000 字节

可存放的表项数为 2000/5=400（个）.

当空白区数目大于 400 时，空白文件目录大于位示图。

12.2 假设一个磁盘驱动器有 5000 个柱面，从 0 到 4999，驱动器正在为柱面 143 的一个请求提供服务，且前面的一个服务请求是在柱面 125。按 FIFO 顺序，即将到来的请求队列是

86，1470，913，1774，948，1509，1022，1750，130

从现在磁头位置开始，按照下面的磁盘调度算法，要满足队列中即将到来的请求要求磁头总的移动距离（按柱面数计）是多少？

a. FCFS b. SSTF c. SCAN d. LOOK e. C-SCAN

答：a. FCFS 的调度是 143 ， 86 ， 1470 ， 913 ， 1774 ， 948 ， 1509 ， 1022 ， 1750 ， 130 。总寻求距离是 7081 。

b. SSTF 的调度是 143 ， 130 ， 86 ， 913 ， 948 ， 1022， 1470， 1509， 1750， 1774。总寻求距离是 1745。

c. SCAN 的调度是 143 ， 913 ， 948 ， 1022， 1470， 1509， 1750， 1774 ， 4999 ， 130 ， 86 。总寻求距离是 9769 。

d. LOOK 的调度是 143 ， 913 ， 948 ， 1022， 1470， 1509， 1750， 1774， 130 ， 86 。总寻求距离是 3319 。

e. C-SCAN 的调度是 143 ， 913 ， 948 ， 1022 ， 1470 ， 1509 ， 1750 ， 1774 ， 4999 ， 86 ， 130 。总寻求距离是 9985 。

f. C-LOOK 的调度是 143 ， 913 ， 948 ， 1022 ， 1470 ， 1509 ， 1750 ， 1774 ， 86 ， 130 。总寻求距离是 3363 。

12.14 MTBF（平均无故障时间）是硬盘可靠性的一个指标。虽然这个指标被称作“时间”，但实际上 MTBF 通常是以设备的正常工作小时数度量的。

(1) 如果一个系统包含 1000 个磁盘驱动器，每个驱动器的 MTBF 是 750000 小时，下面的描述中哪一个最符合该系统发生一次磁盘故障的时间：每 1000 年，每世纪，每十年，每个月，每个星期，每天，每小时，每分钟，每秒钟？

(2) 统计表明，一个 20 到 21 岁的美国公民平均死亡率为千分之一，由此推论 20 岁的 MTBF 时间（单位由小时转换为年），对于一个 20 岁的人来说，MTBF 给出期望的寿命是多大？

(3) 某类磁盘驱动器，生产商保证的 MTBF 为 1 百万小时 你能推算出它们的保质期是多少年吗？

答：

(1) 750000 / 1000=750(小时) 约等于31天，每个月发生一次磁盘故障。

(2) 1年是8760小时，8760小时 / 0.001 ＝ 8760000小时（1000年）也就是说对于一个20岁的人来说，MTBF给出期望的寿命是1000年，这没有任何实际意义。

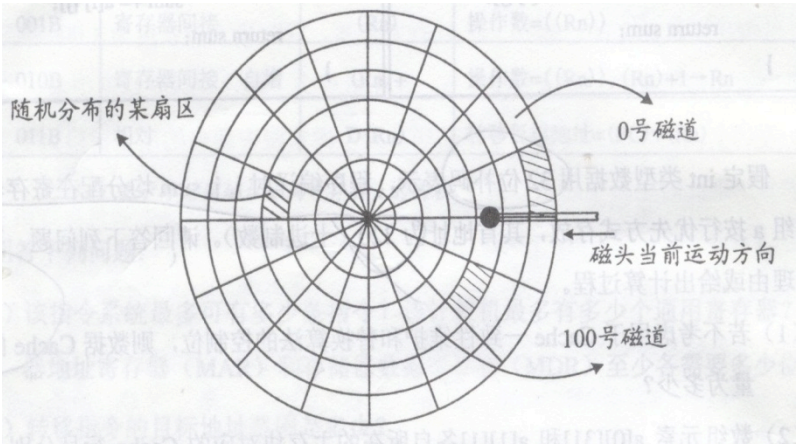
(3) 从上一小题可看出，MTBF给出期望的寿命没有任何实际意义。一般来说，磁盘驱动器设计的寿命是5年，假如真的有一个MTBF为1百万小时的磁盘，那么在其期望的寿命内是不可能有故障的。

12.01 假设计算机系统采用 CSCAN（循环扫描）磁盘调度策略，使用 2KB 的内存空间记录 16384 个磁盘块的空闲状态。

（1） 请说明在上述条件下如何进行磁盘块空闲状态管理。

（2） 设某单面磁盘旋转速度为每分钟 6000 转。每个磁道有 100 个扇区，相邻磁道间的平均移动时间为 1ms。若在某时刻，磁头位于

100 号磁道处，并沿着磁道号增大的方向移动（如下图所示），磁道号请求队列为 50、90、30、120，对请求队列中的每个磁道需读取 1 个随机分布的扇区，则读完这 4 个扇区总共需要多少时间？要求给出计算过程。



（3） 如果将磁盘替换为随机访问的 Flash 半导体存储器（如 U 盘、SSD 等），是否有比 CSACN 更高效的磁盘调度策略？若有，给出磁盘调度策略的名称并说明理由；若无，说明理由。

答：（1）用位图表示磁盘的空闲状态。每一位表示一个磁盘块的空闲状态，共需要 16384/8=2048 字节=2KB。系统提供的 2KB 内存能正好能表示 16384 个磁盘块。

（2）采用 CSCAN 调度算法，访问磁道的顺序为 50、90、30、120，则磁头移动磁道长度为 20+90+20+40=170，总的移动磁道时间为 170×1ms=170ms。

由于转速为 6000 转/分，则平均旋转延迟为（60/6000）/2 s=5ms，要访问 4 个磁道，总的旋转延迟时间为=4×5ms=20ms。

由于转速为 6000 转/分，则读取一个磁道上的一个扇区的平均读取时间为（60/6000）/100 s =0.1ms，总的读取扇区的时间=4×0.1ms=0.4ms。

读取上述磁道上所有扇区所花的总时间=170ms+20ms+0.4ms=190.4 ms

（3）采用 FCFS（先来先服务）调度策略更高效。因为 Flash 半导体存储器的物理结构不需要考虑寻道时间和旋转延迟，可直接按 I/O 请求的先后顺序服务。

13.3 考虑单用户 PC 机上的下列 I/O 操作：

（1）图形用户界面下使用鼠标

（2）在多任务操作系统下的磁带驱动器（假设没有设备预分配）

（3）包含用户文件的磁盘驱动器

（4）使用存储器映射 I/O，直接和总线相连的图形卡

在操作系统中使用缓冲技术，假脱机技术，Cache 技术，或者它们的组合来实现上述操作。实现时使用轮询 I/O 还是中断 I/O？为什么？

答：（1）在鼠标移动时，如果有高优先级的操作产生，为了记录鼠标活动的情况，必须使用缓冲技术，另外，假脱机技术和 Caching 技术不是很必要，而应采用中断驱动 I/O 方式。

（2）由于磁带驱动器和目标或源 I/O 设备间的吞吐量不同，必须采用缓冲技术；为了能对储存在磁带上的数据进行快速访问，必须采用 Caching 技术；当有多个用户需要对磁带进行读或写的时候，假脱机技术也是必须采用的；为了取得最好的性能，应该采用中断驱动 I/O 方式。

（3）为了能使数据从用户作业空间传送到磁盘或从磁盘传送到用户作业空间，必须采用缓冲技术；同样道理，也必须采用 Caching 技术；由于磁盘是属于共享设备，故没必要采用假脱机技术；最好采用中断驱动 I/O 方式。

（4）为了便于多幅图形的存取及提高性能，缓冲技术是可以采用的，特别是在显示当前一幅图形时又要取得下一幅图形时，应采用双缓冲技术；基于存储器映射及直接和总线相连的图形卡是快速和共享设备，所以没必要采用假脱机技术和 Caching 技术；轮询 I/O 和中断 I/O 只对输入和 I/O 是否完成的检测有用，而对于采用存储器映射的设备不必用到上述两种 I/O 方式。

13.01 驱动程序是什么？为什么要有设备驱动程序？用户进程怎样使用驱动程序？

答：（1）用户进程与设备控制器之间的通信程序称为设备驱动程序。

（2）设备驱动程序是控制设备动作的核心模块，如设备的打开、关闭、读、写等，用来控制设备上数据的传输。它直接与硬件密切相关，处理用户进程发出的 I/O 请求。

（3）用户进程使用设备驱动程序时，设备驱动程序的处理过程为：将用户进程抽象的 I/O 要求转换为具体的要求，检查 I/O 请求的合法性，读出和检查设备的状态，传送必要的参数，设置设备工作方式，启动 I/O 设备。