操作系统作业二

班级 F0903028　　学号 5090109228　　姓名 杜若飞

一. 简答题：(7*5)

1. 死锁和竞争有何关系?

死锁由于竞争产生，竞争不一定都会产生死锁。原因如下：

(1) 首先，死锁是多个进程竞争资源造成的一种僵局。若无外力作用，这些进程将永远不能向前推进。

(2) 因此，死锁是由于系统中多个进程所共享的资源不足以同时满足需要时，引起对资源的竞争而产生的。但竞争资源不一定都会产生死锁，因为只要进程调度方案合法，就不会产生死锁。

2. 什么是系统抖动?

在更换页面时，如果更换的页面是一个很快就会被再次访问的页面，则在此次缺页中断后很快又会发生新的缺页中断。这种现象就称为系统抖动或内存抖动或系统颠簸。这种现象会大大降低系统效率。

3．为什么要引入 SPOOLing 系统? SPOOLing 系统可带来哪些好处?

（1） SPOOLing 是一种为了解决系统并发而发明的设备管理技术

所有字符设备本质上属于顺序存取设备，都是独享设备并属于慢速设备，。因此，一个进程在某台字符设备上进行数据交换时，往往要等待较长时间，并且在该数据交换完成之前，其他进程不能同时访问这台设备。而且动态分配也不能真正提高这类设备的利用率，当一个进程正在使用这类设备进行一次较大量的数据交换时，其他需要同时访问该设备的进程就要等待较长的时间，从而降低了整个系统的并发能力。SPOOLing 技术正是针对上述问题提出的一种设备管理技术。

SPOOLing 系统的核心思想是利用一台可共享、高速、大容量的块设备(磁盘)来模拟独占设备的操作，使一台独占设备变成多台可并行使用的虚拟设备。SPOOLing 系统主要由输入井和输出井、输入缓冲区和输出缓冲区、输入进程和输出进程三部分组成。

（2）SPOOLing 系统可带来的好处如下:

提高了 I/O 操作的速度，将独占设备改造为共享设备，实现了虚拟设备的功能。具体来说：

和各虚拟设备之间的数据交换由 SPOOLing 进程统一调度实施，而且这种数据交换以并行方式进行，系统呈现出高度的并行性
用户使用的是虚拟设备，可以减少用户进程的等待时间
在多道程序系统中，用程序模拟脱机输入/输出时外围控制机的功能，这样便可在主机的直接控制下实现脱机输入/输出功能。此时的外围操作与 CPU 对数据的处理同时进行，这种在联机情况下实现的外围设备同时操作称为 SPOOLing，也称伪脱机。

4．覆盖技术与虚拟存储技术有何本质不同?交换技术与虚存中使用的调入/调出技术有何相同与不同之处?

（1）

类别	覆盖技术	虚拟存储技术
覆盖的程序段的最大长度【最本质】	受到物理内存容量的限制	只受计算机地址结构的限制
对用户程序要求	要求程序员必须精心地设计程序及其数据结构，使得要覆盖的段具有相对独立性，不存在直接联系或相互交叉访问。	对用户的程序段没有特殊要求

（2）

	交换技术	虚存中使用的调入/调出技术
相同点	在内存与外存之间交换信息	
主要区别	换进换出整个进程（进程结构和共享正文段除外），因此一个进程的大小受物理存储器的限制	在内存和外存之间来回传递的是存储页或存储段,而不是整个进程,从而使得进程的地址映射具有了更大的灵活性,且允许进程的大小比可用的物理存储空间大得多。

5.关于处理机调度，试问：

（1）什么是处理机三级调度？
（2）处理机三级调度分别在什么情况下发生？
（3）各级调度分别完成什么工作？

（1） 操作系统中有三级调度：高级调度(作业调度)、中级调度(交换调度)和低级调度(进程调度)。它们构成系统内的多级调度。不同类型的操作系统不一定完全都实现上述三种调度。

（2） 处理机三级调度发生的情况是:

高级调度	根据系统内所有资源的使用情况，一旦可能便从后备作业中选择一道作业进入系统，并创建相应的进程，分配必要的系统资源，然后将进程"就绪"。
低级调度(CPU 调度)	根据 CPU 资源的使用情况及时分配 CPU， 即从"就绪"的进程中选择一个进程在 CPU 上"运行"。这种调度不仅要求调度算法本身的时间复杂度小，而且要求策略精良，因为低级调度直接影响着系统的整体效率。在多道程序系统中必须提供低级调度。
中级调度	在内存中常常有许多进程处于某种等待状态，这些进程在"等待"期间无谓地占用着内存资源。如将它们暂时换至外存，则所节省出来的内存空间可用以接纳新的进程，一旦换出外存的进程，具备运行条件时再将其重新换入内存。为此，在逻辑上将主存延伸，用一部分外存空间(称为交换区)替代主存，并且实施交换调度(中级调度)。在各种类型的操作系统中可以根据内存的配置、系统能承受的最大负载，有选择地进行中级调度，或者不实施中级调度。

（3）

高级调度	完成作业调度，后备"状态的作业变为"执行"状态
中级调度	完成内存和外存信息的交换调度
低级调度	完成进程调度，使"就绪"的进程在 CPU 上"运行"

6.固定分区管理、可变分区管理、页式管理、段式管理、段页式管理各会产生何种碎片？

固定分区管理	内碎片	固定式分区把内存划分为若干个固定大小的连续分区，但产生内碎片造成浪费。
可变分区管理	外碎片	动态分区在装入程序时按其初始要求分配，或在其执行过程中通过系统调用进行分配或改变分区大小，引入了外碎片。
页式管理	内部碎片	每个内碎片不超过页大小。
段式管理	外部碎片	没有内碎片，外碎片可以通过内存紧缩来消除。
段页式管理	基本无碎片	每段所拥有的程序和数据在内存中可以分开存放，分段的大小也不再受内存可用区的限制。

7.消息缓冲通信技术是一种高级通信机制，由 Hansen 首先提出。

（1）试述高级通信机制与低级通信机制 P、V 原语操作的主要区别。

	高级通信机制	低级通信机制 P、V 原语
交换信息量	解决了进程间的同步互斥问题，能很好交换大量消息。	进程间只能交换一些信息，基本上只能是控制信息，缺乏传输消息的能力。
用户编程复杂性	高效传输大量信息，且操作系统隐藏了进程通信的实现细节，即通信过程对用户是透明的。这样就大大降低了编程复杂度。	在程序中增加 P、V 原语编程，不但增加了编程的复杂性，降低了程序直观性，同时由于编程不当，有可能出现死锁，难以查找其原因。

（2）请给出消息缓冲机制(有界缓冲)的基本原理。

所谓消息(Message)，是指一组信息，消息缓冲区是含有如下信息的缓冲区:

指向发送进程的指针	Sptr
指向下一信息缓冲区的指针	Nptr
消息长度	Size
消息正文	Text

消息缓冲通信机制的基本工作原理是

把消息缓冲区作为进程通讯的一个基本单位，为了实现进程之间的通讯，系统提供了发送原语 Send(A)和接收原语 Receive(B)。

每当发送进程欲发送消息时，发送进程用 Send(A)原语把欲发送的消息从发送区复制到消息缓冲区，并将它挂在接收进程的消息队列末尾。

如果该接收进程因等待消息而处于阻塞状态，则将其换醒。而每当接收进程欲读取消息时，就用接收原语 Receive(B)从消息队列头取走一个消息放到自己的接收区。

(3) 消息缓冲通信机制 (有界缓冲) 中提供发送原语 Send(receiver，a)，调用参数 a 表示发送消息的内存区首地址，试设计相应的数据结构，并用 P、V 原语操作实现 Send 原语。

消息缓冲通信机制中，消息队列属于临界资源，故在 PCB 中设置了一个用于互斥的信号量 mutex，而每当有进程要进入消息队列时，应对信号量 mutex 施行 P 操作，退出消息队列后，应对信号量 mutex 施行 V 操作。由于接受进程可能会收到几个进程发来的消息，故应将所有的消息缓冲区链成一个队列，其队头由接收进程 PCB 中的队列头指针 HeadPointer 指出。

为了表示队列中的消息的数目，在 PCB 中设置了信号量旬，每当发送进程发来一个消息，并将它挂在接收进程的消息队列上时，便在 Sn 上执行 V 操作:而每当接收进程从消息队列上读取一个消息时，先对 Sn 执行 P 操作，再从队列上移出要读取的消息。

用 P、V 原语操作实现 Send 原语和 Receive 原语的处理流程如下

```
void send(receiver, msg)              //发送原语
{
    getBuffer(msg, size, sendMsg);          //申请消息缓冲区
    sendMsg.sender = msg.sender;            //将发送区的信息发送到消息缓冲区
    sendMsg.size = msg.size;
    sendMsg.text = msg.size;
    sendMsg.next = 0;

    getId(PCB set, receive, revMsg);    //获得接收进程的内部标识符
    P(revMsg.mutex);
    insert(revMsg.head, sendMsg);           //消息缓冲区插入到消息队列首
    V(revMsg.Sn);
    V(revMsg.mutex);
}

void receive(msg)                           //接收原语
{
    Pid receiver;                           //接收进程内部标识符
    P(receiver.Sn);
    P(receiver.mutex);
    rev = receiver.removeFirst()    //从消息队列中移出第一个消息
    V(receiver.mutex);
    msg.sender = rev.sender;
    b.size = rev.size;                      //将消息缓冲区中的信息复制到接收区
    b.text = rev.text;
}
```

二．应用题：(3*8)

8.有一矩阵 Var A:array[1..100，1..100] of integer: 以行为先进行存储。有一个虚存系统，物理内存共有三页，其中一页用来存放程序， 其余两页用于存放数据。假设程序已在内存中占一页，其余两页空闲。

```
程序 A:
        for   i:=1 to 100 do
            for   j:=1 to 100 do
                  A[i，j]:=0;
程序 B:
    for j:=1 to 100 do
        for i:=1 to 100 do
            A[i，j]:=0;
```

若每页可存放 200 个整数，程序 A 和程序 B 的执行过程各会发生多少次缺页?

试问:若每页只能存放 100 个整数呢? 以上说明了什么问题?

1. 每个主存块的大小为可以存放 200 个数组元素，有两个内存块可以用来存放数组信息，数组中的元素按行编址。
i)对于程序 A，数组访问顺序是:

A[1，1]，A[1，2]，A[1，3]，……，A[1，100]
A[2，1]，A[2，2]，A[2，3]，……，A[2，100]
A[100，1]，A[100，2]，A[100，3]，……，A[100，100]

显然，数组的存储顺序与访问顺序一致，每访问两行数组遇到一次缺页中断。如果采用 LRU 页面调度算法，

程序 A 会产生 50 次缺页中断。

对于程序 B，数组访问顺序是:

A[1，1]，A[2，1]，A[3，1]，……，A[100，1]
A[1，2]，A[2，2]，A[3，2]，……，A{100，2]
A[1，100]，A[2，100]，A[3，100]，……，A[100，100]

显然，数组的存储顺序(行序)与访问顺序(列序)不一致，每访问两个数组元素遇到一次缺页中断。如果采用 LRU 页面调度算法，

程序 B 会产生 5000 次缺页中断。

若每页只能存放 100 个整数，对于程序 A，数组的存储顺序与访问顺序一致，每访问一行数组遇到一次缺页中断。如果采用 LRU 页面调度算法，会产生 100 次缺页中断。

对于程序 B，数组的存储顺序(按行的顺序)与访问顺序(按列的顺序)不一致，每访问一个数组元素遇到一次缺页中断。如果采用 LRU 页面调度算法，会产生 10000 次缺页中断。

以上结果说明：页面越大，缺页中断次数越少；页面越小，缺页中断次数越多。

9. 在一个分页存储管理系统中,页面大小为 4KB,系统中的地址占 24 位,给定页面变换表如下表所示。

页号 P	块号 B
0	3
1	4
2	9
3	7

(1)计算逻辑地址(页号为 3,页内地址为 100)的物理地址:
(2)说明地址变换过程。

(1) 逻辑地址(页号为 3，页内地址为 100)的物理地址为:

$$7 \times 4KB + 100 = 28KB + 100 = 28772$$

(2) 在请求分页存储管理方案中，系统是通过页面变换表来进行地址转换的。先将逻辑地址分解成页号 P 和页内地址 W 两部分，然后查页面变换表，可得页号 P 对应的物理块号为 B，从而变换出对应的物理地址为:

$$物理地址 = 块号 \times 页面大小 + 页内地址$$

10. 设系统中有三种类型的资源(A、B、C)和五个进程(P1、P2、P3、P4、P5)，A 资源的数量为 17，B 资源的数量为 5，C 资源的数量为 20。在 T0 时刻系统状态如表 1 和表 2 所示。系统采用银行家算法实施死锁避免策略。

表 1　T_0时刻系统状态

进程	最大资源需求量			已分配资源数量		
	A	B	C	A	B	C
P_1	5	5	9	2	1	2
P_2	5	3	6	4	0	2
P_3	4	0	11	4	0	5
P_4	4	2	5	2	0	4
P_5	4	2	4	3	1	4

表 2　T_0时刻系统状态

	A	B	C
剩余资源数	2	3	3

①T0 时刻是否为安全状态？若是，请给出安全序列。

T_0时刻是安全状态，安全序列为(P_4，P_5，P_1，P_2，P_3)。

②在 T0 时刻若进程 P2 请求资源(0，3，4)，是否能实施资源分配？为什么？

不能分配。因为所剩余的资源数量不够。

③在②的基础上，若进程 P4 请求资源(2，0，1)，是否能实施资源分配？为什么？

可以分配。当分配完成后，系统剩余的资源向量为(0，3，2)，这时仍可找到一个安全序列，

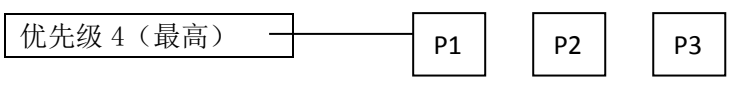
(P_4，P_5，P_1，P_2，P_3)。

④在③的基础上，若进程请求资源(0，2，0)，是否能实施资源分配？为什么？

不能分配。若分配完成后，系统剩余的资源向量为(0，3，0)，这时无法找到一个安全的序列。

四．综合题：

11. （8 分）图 1 中将一组进程分为 4 类，各类进程之间采用优先级调度，而各类进程内部采用时间片轮转调度，请简述 P1，P2，P3，p4，P5，P6，p7，P8 进程的调度过程。



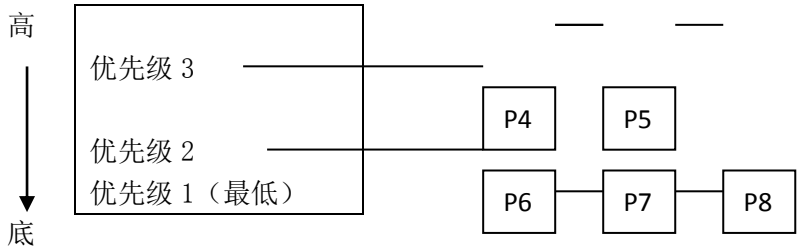


图 1

（1）先进行优先级 4 的进程调度，P1，P2，P3 按时间片进行轮转；

（2）等 P1，P2，P3 均执行完毕，执行优先级 3 的进程 P4，P5。

（3）等 P4，P5 按时间片轮转完毕后，调度优先级 1 的进程 P6，P7，P8。

（4）进程 P6，P7，P8 按时间片轮转直至完成。

12. (8 分) 从读卡机上读进 N 张卡片，然后复制一份，要求复制出来的卡片与读进来的卡片完全一致。这一工作由三个进程 get，copy 和 put 以及两个缓冲区 buffer1 和 buffer2 完成。进程 get 的功能是把一张卡片上的信息从读卡机上读进 buffer1；进程 copy 的功能是把 buffer1 中的信息复制到 buffer2；进程 put 的功能是取出 buffer2 中的信息并从行式打印机上打印输出。试用 P、V 操作完成这三个进程间的尽可能并发正确运行的关系(用程序或框图表示)，并指明信号量的作用及初值。

```
    void init() {
        Semaphore acc1(1), acc2(1);     // 互斥信号量acc1，acc初值为1，分别用于对buffer1
和buffer2的互斥访问;
        Semaphore idle1(1), idle2(1);   // 同步信号量idle1，idle2初值为1，分别表示bufferl
和buffer2初始状态为空闲，可以放一张卡片信息;
        Semaphore msg1(0), msg2(0);     // 同步信号量msg1，msg2初值为0，分别表示buffer1
和buffer2中的信息还没有被取用(或已被取用了)。
    }
    void get() {
        while (true){
            if (receiveSocket.available()){        //读卡机有信息
```

```
            message = receiveSocket.readBytes();
            idle1.P(); acc1.P();
            buffer1 = message;
            msg1.V(); acc1.V();
        }
    }
}
void copy(){
    while (true){
        acc1.P();  msg1.P();
        message = buffer1;
        idle1.V(); acc1.V();
        idle2.P(); acc2.P();
        buffer2 = message;
        msg2.V(); acc2.V();
    }
}
void put(){
    while (true){
        acc2.P(); msg2.P();
        message = buffer2;
        idle2.V(); acc2.V();
        printf(message);
    }
```

}

13. (8 分) 在一个请求分页存储系统中，一个程序的页面走向为 4、3、2、1、4、3、5、4、3、2、1、5，设分配给该程序的存储块数为 4，试分别求出采用 FCFS 和 LRU 页面置换算法发生缺页中断的次数和缺页中断率（要求给出页面置换的过程）。

（1）FCFS 算法的页面置换如下：

时刻	1	2	3	4	5	6	7	8	9	10	11	12
页面走向	4	3	2	1	4	3	5	4	3	2	1	5
M=4	4+	3+ 4	2+ 3 4	1+ 2 3 4	1 2 3 4	1 2 3④	5+ 1 2 ③	4+ 5 1 ②	3+ 4 5 ①	2+ 3 4 ⑤	1+ 2 3 ④	5+ 1 2 3
标志	+	＋	＋	＋			＋	＋	＋	＋	＋	＋

缺页次数

$$F = 10$$

缺页中断率

$$\frac{10}{12} \times 100\% = 83.3\%$$

（2）LRU 算法的页面置换如下：

时刻	1	2	3	4	5	6	7	8	9	10	11	12
页面走向	4	3	2	1	4	3	5	4	3	2	1	5
M=4	4+	3+ 4	2+ 3 4	1+ 2 3 4	4 1 2 3	3 4 1②	5+ 3 4 1	4 5 1 1	3 4 5 1	2+ 3 4 ⑤	1+ 2 3 ④	5+ 1 2 3
标志	+	＋	＋	＋			＋			＋	＋	＋

缺页次数

$$F = 8$$

缺页中断率

$$\frac{8}{12} \times 100\% = 66.7\%$$

14.（17 分）请求分页管理系统中，假设某进程的页表内容如下表所示。

页号	页框号	有效位 （存在位）

页面大小为 4KB，一次内存的访问时间是 100ns，一次快表（TLB）的访问时间是 10ns，处理一次缺页的平均时间为 108ns（已含更新 TLB 和页表的时间），进程的驻留集大小固定为 2，采用最近最少使用置换算法（LRU）和局部淘汰策略。假设

0	**101H**	**1**
1	**--**	**0**
2	**254H**	**1**

①TLB 初始为空；
②地址转换时先访问 TLB，若 TLB 未命中，再访问页表
（忽略访问页表之后的 TLB 更新时间）；
③有效位为 0 表示页面不在内存，产生缺页中断，缺页中断处理后，返回到产生缺页中断的指令处重新执行。设有虚地址访问序列
2362H、1565H、25A5H，请问：
（1） 依次访问上述三个虚地址，各需多少时间？给出计算过程。
（2） 基于上述访问序列，虚地址 1565H 的物理地址是多少？请说明理由。

（1） 根据页式管理的工作原理，应先考虑页面大小，以便将页号和页内位移分解出来。
页面大小为 4KB，即2^{12}，则得到页内位移占虚地址的低 12 位，页号占剩余高位。
可得三个虚地址的页号如下
（十六进制的一位数字转换成 4 位二进制，因此，十六进制的低三位正好为页内位移，最高位为页号）：

2362H：页号为 2，访问 TLB 用时 10ns，因初始为空，再访问页表 100ns 得到页框号，合成物理地址后访问主存 100ns，共计

$$10ns + 100ns + 100ns = 210ns$$

1565H：页号为 1，访问 TLB 用时 10ns，落空，访问页表 100ns 落空，进行缺页中断处理 108ns，合成物理地址后访问主存 100ns，共计

$$10\text{ns} + 100\text{ns} + 108\text{ns} + 100\text{ns} \approx 108\text{ns}$$

25A5H：页号为 2，访问 TLB，因第一次访问已将该页号放入 TLB，因此花费 10ns 便可合成物理地址，访问主存 100ns，共计

$$10\text{ns} + 100\text{ns} = 110\text{ns}$$

（2）当访问虚地址 1565H 时，产生 Page fault，合法驻留集为 2，必须从页表中淘汰一个页面，根据题目的置换算法，应淘汰 0 号页面，因此 1565H 的对应页框号为

$$101\text{H}$$

由此可得虚地址 1565H 的物理地址为

$$101565\text{H}$$