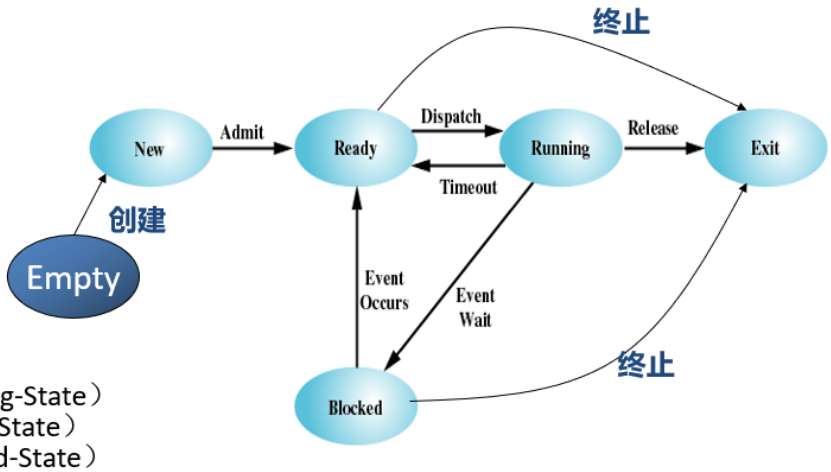
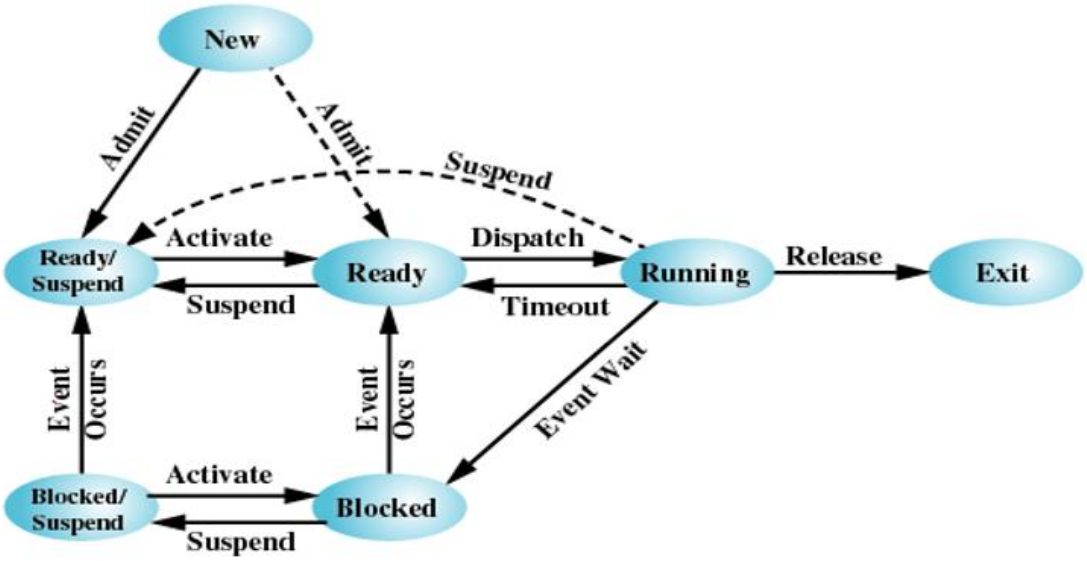
操作系统要点提要

1. 并发和共享是 OS 两个最基本的特性，它们又是互为存在条件。一方面资源共享是以程序（进程）的并发性执行为条件的，若系统不允许程序并发执行，自然不存在资源共享问题。另一方面若系统不能对资源共享实施有效管理，则也必将影响到程序并发执行。
2. 异常引起中断，中断不一定是由异常引起的。
3. PCB：进程标识、处理器状态信息、进程控制信息
4.



运行态（Running-State）
就绪态（Ready-State）
阻塞态（Blocked-State）
新建态（New-State）
退去态（Exit-State）

5.



6. 模式切换不同于进程切换，它并不引起进程状态变化，也不一定引起进程的切换，在完成了中断调用之后，完全可以再通过一次逆向的模式切换来继续执行用户进程。
7. UNIX fork() 返回子进程标识符给父进程，把 0 值返回给子进程。
8. 在一个已有进程中创建一个新线程比创建一个全新进程所需要的时间要少许多。终止一个线程比终止一个进程花费的时间少。同一进程内线程间的切换比进程间切换花费的时间少。线程提高了不同的执行程序间的通信的效率，无需内核参与。

9. 内核级线程，进程内线程切换需要模式切换（调度：内核模式，执行：用户模式）。用户级线程，系统调用引发阻塞，一个线程执行一个系统调用时，不仅这个线程会被阻塞，进程中的所有线程都会被阻塞。

10. 用 P、V 操作实现下述问题。桌子上有一个盘子，可以存放一个水果，父亲总是放苹果到盘子中，而母亲总是放香蕉到盘子中；一个儿子专等吃盘中的香蕉，而一个女儿专等吃盘中的苹果。

设 dish=1 表示盘子是否为空；apple=0 表示盘中是否有苹果；banana=0 表示盘中是否有香蕉。

```
    semaphore dish=1;
    semaphore apple=0;
    semaphore banana=0;
    Main()
       {   parbegin( father(),     mother(),
                 son(),   daughter());
           }
Father()
     {   while(true)
          {        p(dish);
                   将苹果放入盘中；
                   v(apple);
            }
        }
Mother()
       {    while(true)
             { p(dish);
               将香蕉放入盘中；
                    v(banana);
             }
       }
son()
        {   while(ture)
             {        p(banana);
             从盘中取出香蕉；
                      v(dish);
                     吃香蕉；
             }
        }
daughter()
        {   while(ture)
             {        p(apple);
                      从盘中取出苹果；
                      v(dish);
                     吃苹果；
             }
```

}

11. 在一个盒子里，混装了数量相等的黑白围棋子。现在利用自动分拣系统把黑子、白子分开，设分拣系统有两个进程 P1 和 P2，其中进程 P1 拣白子；进程 P2 拣黑子。规定每个进程一次拣一子，当一个进程在拣时不允许另一个进程去拣，当一个进程拣了一子时，必须让另一个进程去拣。试写出进程 P1 和 P2 能够正确并发执行的程序。

设私有信号量 S1=1,S2=0;

```
P1(){                          P2(){
    P(S1);                         P(S2);
    拣白子;                        拣黑子;
    V(S2);}                        V(S1);}
```

12. 用 P、V 操作管理临界区时，互斥信号量的初值应定义为(　)。

A．任意值

B．1

C．0

D．-1

13. 用 PV 操作管理互斥使用的资源时，信号量的初值应定义为（ ）。

A．任意正整数

B．1

C.　0

D. -1

14.

```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1, n= 0, e= sizeofbuffer;
void producer()
{
     while (true) {
          produce();
          semWait(e);
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          semSignal(e);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

15. 死锁预防

占有且等待

要求进程一次性请求所需要的资源

所有资源都满足时才执行

缺点：

进程不需要所有资源也可继续执行

资源被占用但长时间不用

进程未来所需要的资源未知

非抢占

申请被拒绝，主动释放已分配的其它资源

申请被拒绝，抢占已分配给其它进程的该资源

循环等待

资源的线性顺序

只能申请已有资源后面的资源

16. 死锁避免

允许三个必要条件

动态选择确保不会达到死锁点

两种方法：

如果一个进程的请求会导致死锁，则不启动此进程（进程启动拒绝）

如果一个进程增加的资源请求会导致死锁，则不允许此分配（资源分配拒绝）

17. 死锁检测

死锁恢复

取消所有的死锁进程

把每个死锁进程回滚到前面定义的某些检查点，并且重新启动所有进程

可能会再次发生死锁

连续取消死锁进程直到不再死锁

连续抢占资源直到不再存在死锁

18. 哲学家就餐

```
/* program diningphilosophers */
semaphore fork[5] = {1};
semaphore room = {4};
int i;
void philosopher (int I)
{
    while (true)
    {
     think();
     wait (room);
     wait (fork[i]);
     wait (fork [(i+1) mod 5]);
     eat();
     signal (fork [(i+1) mod 5]);
     signal (fork[i]);
     signal (room);
    }

}
void main()
{
    parbegin (philosopher (0), philosopher (1), philosopher (2),
          philosopher (3), philosopher (4));
}
```

Figure 6.13 A Second Solution to the Dining Philosophers Problem

19.

T0 时刻若出现下述资源分配情况：

process	allocation			Max/claim			available		
	A	B	C	A	B	C	A	B	C
P1	0	2	2	1	4	6	2	1	4
P2	3	0	2	7	2	4			
P3	3	0	3	6	5	10			
P4	9	2	0	10	5	5			
P5	1	1	0	2	2	3			

试问：（1）T0 时刻，P1 请求 Request1(1,0,1)，系统能否分配资源，为什么？
（2）T0 时刻，P3 请求 Request3(1,0,1)，系统能否分配资源，为什么？

参考答案
（1）T0 时刻，P1 请求 Request1(1 ,0,1)，系统能否分配资源，为什么？
Request1(1 ,0,1)<= (1,2,4) //小于 Need1,请求合理
Request1(1 ,0,1)<= (2,1,4)//小于 available,系统可满足
系统假设分配，修改数据

process	allocation			Max/claim			Need			available		
	A	B	C	A	B	C	A	B	C	A	B	C
P1	1	2	3	1	4	6	0	2	3	1	1	3
P2	3	0	2	7	2	4	4	2	2			
P3	3	0	3	6	5	10	3	5	7			
P4	9	2	0	10	5	5	1	3	5			
P5	1	1	0	2	2	3	1	1	3			

找安全的资源分配序列：

process	Work			Need			Allocation			Work+ Allocation			Finish
P5	1	1	3	1	1	3	1	1	0	2	2	3	T
P1	2	2	3	0	2	3	1	2	3	3	4	6	T
P4	3	4	6	1	3	5	9	2	0	12	6	6	T
P2	12	6	6	4	2	2	3	0	2	15	6	8	T
P3	15	6	8	3	5	7	3	0	3	18	6	11	T

存在安全序列 P5，P1，P4，P2，P3，分配后系统依然处于安全状态，可以分配

（2）T0 时刻，P3 请求 Request3(1,0,1)，系统能否分配资源，为什么？
Request3(1 ,0,1)<= (3,5,7) , Request3(1 ,0,1)<= (2,1,4)，系统假设分配，修改数据

process	allocation			max			Need			available		
	A	B	C	A	B	C	A	B	C	A	B	C
P1	0	2	2	1	4	6	1	2	4	1	1	3
P2	3	0	2	7	2	4	4	2	2			
P3	4	0	4	6	5	10	2	5	6			
P4	9	2	0	10	5	5	1	3	5			
P5	1	1	0	2	2	3	1	1	3			

找安全的资源分配序列：

process	Work			Need			Allocation			Work+ Allocation			Finish
P5	1	1	3	1	1	3	1	1	0	2	2	3	T
P1	2	2	3	1	2	4							F
P2	2	2	3	4	2	2							F
P3	2	2	3	2	5	6							F
P4	2	2	3	1	3	5							F

无法找到安全序列，分配后系统会不安全。不能分配，系统阻塞进程 P3。

20. 固定分区，内部碎片 动态分区，外部碎片

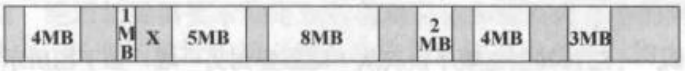
21.

假设使用动态分区，下图是经过数次放置和换出操作后的内存格局。内存地址从左到右增长；灰色区域是分配给进程的内存块；白色区域是可用内存块。最后一个放置的进程大小为 2MB，用 X 标记。此后仅换出了一个进程。

a. 换出进程的最大尺寸是多少？

b. 创建分区并分配给 X 之前，空闲块的大小是多少？

c. 下一个内存需求大小为 3MB。在使用最佳适配/首次适配/下次适配/最差适配的情况下，分别在图上标记出分配的内存区域。



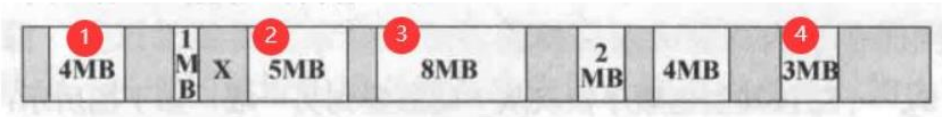
a. 当 2MB 大小的进程被放置进入内存时，它会使用最左边的空闲块。在图中，X 的左边还有一个 1MB 的空闲块，这必定是由 X 放置进入内存之后换出的进程造成的，因而换出进程的最大尺寸是 1MB。

b. X 进程的尺寸+右边邻接的空闲空间=7MB

c.

注：所有标号均代表靠左大小为 3MB 的内存块。

①首次适配 ②下次适配 ③最差适配 ④最佳适配



22.

7.12 系统使用简单分页，内存大小为 2^{32} 字节，页大小为 2^{10} 字节，逻辑地址空间包含 2^{16} 页。

a. 逻辑地址有多少位？

b. 一个页框有多少字节？

c. 物理地址中的多少位是页框号？

d. 页表中有多少表项？

e. 假设每个页表项中含一位有效位，每个页表项有多少位？

a. 逻辑地址空间大小=页大小 x 逻辑地址空间页数=2^{26}，所以逻辑地址应该有 26。

b. 页框大小=页大小，所以一个页框有 2^{10} 字节。

c. 物理地址大小=内存大小/页大小=2^{22}，所以物理地址中有 22 位是页框号。

d. 逻辑地址空间中每一页对应一个表项，所以有 2^{16} 个表项。

e. 由 c，再包含一位有效位，所以每个页表项有 23 位。

23.

系统抖动

如果一个内存块需要被其它进程使用，该块的进程内容就需要被交换出去

处理器大部分的时间都用于交换块，而不是执行指令

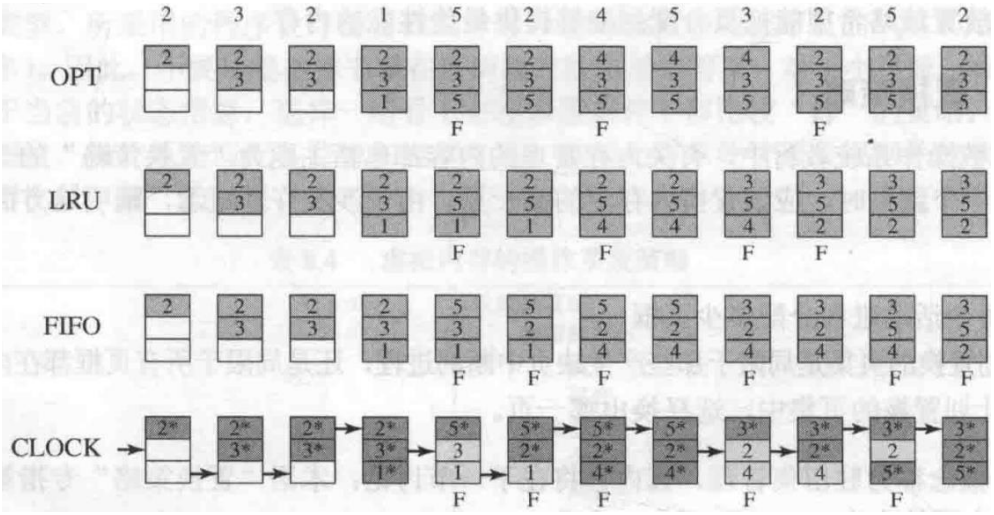
局部性原理

一个进程中的代码和数据的访问有集簇的倾向

因此，假设在很短时间内，仅需要访问进程的一部分块是合理的

对未来可能使用到的块进程预测□避免系统抖动

24. 页缓冲置换算法



FIFO belady 现象

25. 短程调度——低级调度最频繁

SPN（最短进程优先）改进->HRRN

SRT 最短剩余时间 抢占

HRRN 等待时间/服务时间 高优先

周转时间 = 完成时间 – 到达时间

带权： /服务时间

26. 假定，一块数据从外部设备输入到内存所花费的时间为 T，在内存中移动所花费的时间为 M，被用户进程加工处理所花费的时间为 C，那么在使用单 I/O 缓冲区的情况下，平均每块数据的处理时间近似为：max(T,C)+M。当数据从缓冲区复制到用户进程空间时，输入设备不必等待，可立即开始向另一个缓冲区输入数据。因此，增加了一个缓冲区后，前述的平均工作时间可近似为：max(T,C)。

27.

a

FIFO		SSTF		SCAN		C-SCAN	
访问磁道	横跨磁道	访问磁道	横跨磁道	访问磁道	横跨磁道	访问磁道	横跨磁道
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
平均	61.8	平均	29.1	平均	29.6	平均	38

b

SCAN		C-SCAN	
访问磁道	横跨磁道	访问磁道	横跨磁道
110	10	110	10
120	10	120	10
129	9	129	9
147	18	147	18
186	39	186	39
64	122	10	176
41	23	27	17
27	14	41	14
10	17	64	23
平均	29.1	平均	35.1

28. 文件组织形式：堆、顺序文件、索引文件、索引顺序文件、直接文件或散列文件
29. 目录包含有关文件的信息：属性、位置、所有权
30. 三种组块方法：固定组块、可变长度跨越式组块、可变长度非跨越式组块
31. 文件的分配方法：连续分配、链接分配、索引分配
32. 请分别解释在连续分配方式、链接分配方式和索引分配方式中如何将文件的字节偏移量 3500 转换为物理块号和块内偏移量（设盘块大小为 1KB，盘块号需占 4 个字节）。

解：3500/1024 得商为 3，余数为 428，则逻辑块号为 3，块内偏移量为 428。

（1） 在连续分配中，可从相应文件的 FCB 中得到起始物理盘块号，例如 a0，则所求的物理盘块号为 a0+3，块内偏移量为 428

（2） 在链接分配中，由于每块需留 4 个字节存放下一个盘块号，因此逻辑块号为 3500/1020 的商 3，块内偏移为 440。从 FCB 中可得该文件的首个（即第 0 个）盘块的块号，如 b0；然后可从 b0 块得到第 1 个盘块号，如 b1；再从 b1 得到第 2 个盘块号，如 b2；从 b2 得到第 3 个盘块号，如 b3；如此可得所求物理盘块号 b3，块内偏移量为 440。

（3） 在索引分配中，可从文件的 FCB 中得该文件的索引块的地址；再从索引块的第 3 项（距离索引块首字节 12 字节的位置）可获得字节偏移量 3500 对应的物理块号，而块内偏移为 428。

33. 设每个块大小为 4k，一索引项（盘块号）占 8 字节，则
1)直接地址块 iadd(0)-iadd(11)：12 个盘块，小文件（<=48k)则立即读出。
2)一次间址块 iadd(12)：一次间址块中存放 512 个盘块，2M 大小
3)二次间址块 iadd(13)：512*512 个盘块，1G
4)三次间址块 iadd(14): 512*512*512 个盘块，512GB
34. 在 UNIX System V 中，块大小为 4KB，盘块号需 4 个字节.使用索引节点方案，一个文件的最大尺寸是多少：
盘块大小为 4KB，盘块号需 4 个字节
直接地址： 4KB*10=40KB
一级间接地址：4KB*(4KB/4)=4MB
二级间接地址： 4KB* (4KB/4)* (4KB/4)=4GB
三级间接地址：4KB* (4KB/4)* (4KB/4)*(4KB/4) =4TB