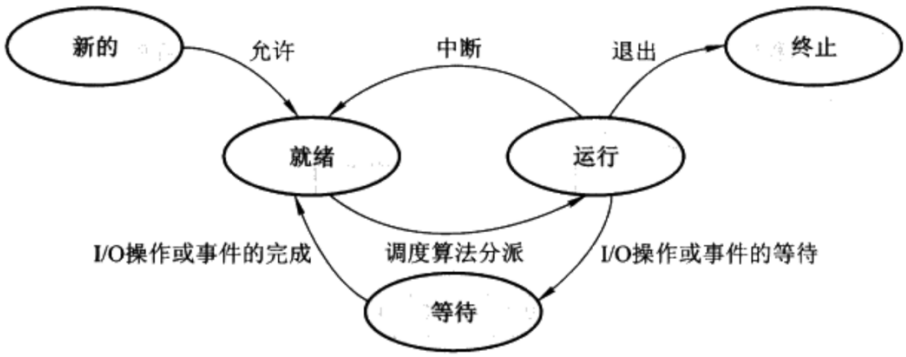
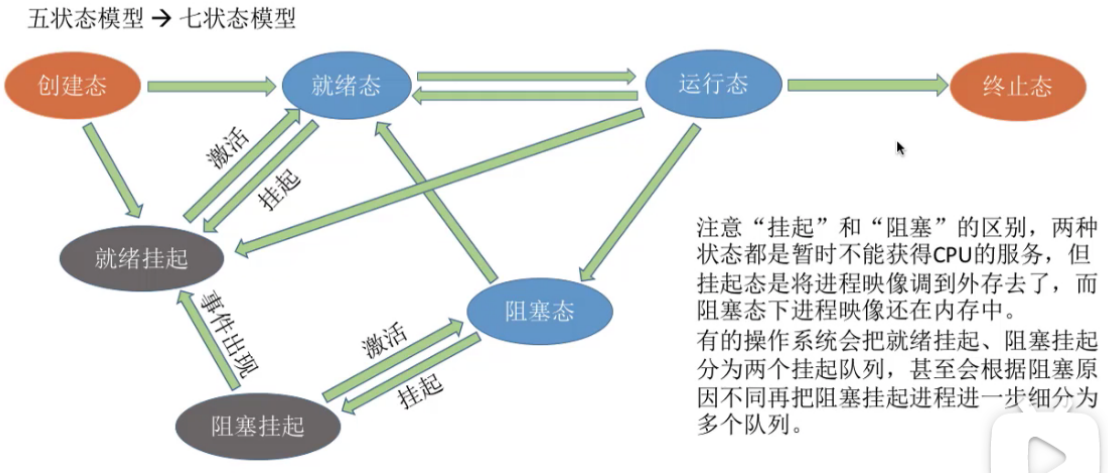
1. 对 CPU、内存、外设并发的操作系统有哪些措施？
2. 根据操作系统对资源和进程的管理，写出中断有哪些方面的作用。
   中断分为内中断（异常/陷阱）和外中断。外中断指来自 CPU 执行指令以外的事情发生，包括外设请求或人为干预，例如 I/O 结束中断、时钟中断。内中断指源自 CPU 执行指令内部的事件，包括程序出错或系统调用。发生中断时，运行用户态的 CPU 会立即进入核心态，提供服务。
3. 描述系统调用的工作机制及其参数传递方法。
   系统调用把应用程序的请求传给内核，每个系统调用和一个数相关联，通过索引表找到相应的内核函数，调用内核函数完成所需的处理，将处理结果返回给应用程序。
   （1）通过**寄存器**来传递参数（2）将参数存在**内存的块和表**中，将块的地址通过寄存器来传递。
   （3）参数通过程序压入**堆栈**，操作系统弹出。
4. 画出进程 NEW、READY、RUNNING、WAITING、TERMINATED 的状态图，并说明状态之间变换的原因。



5. 程序从外存调入内存，在调入、执行、结束过程中发生了什么，又是怎么解决的。



6. 用户级线程和内核级线程是什么？相对的各自有什么优点？
   用户级线程仅存在于用户空间中。对于这种线程的创建、撤销、切换、线程之间的同步与通信功能，都无需利用系统调用来实现，而是通过用户级线程库来实现。线程管理在**用户空间**进行，**效率较高**。
   内核级线程有操作系统**直接支持和管理**，应用程序没有进行线程管理的代码，只有一个到内核级线程的编程接口。当一个线程被阻塞后，允许另一个线程继续执行，所以**并发能力强**。

7. 多队列调度算法和多级反馈队列调度算法的基本思想，比较这两个算法的好坏。
   多队列调度：多级队列调度算法将**就绪**队列分成多个独立队列，根据进程的属性，如内存大小、

进程优先级、进程类型，一个进程被**永久地分配**到一个队列。每个队列都有自己的**调度算法**。此外，队列之间通常采用**固定优先级抢占调度**，每个队列与更低队列相比都有绝对的优先级。或者在队列之间**划分时间片**。优点是**低调度开销**，缺点是**不够灵活**。

多级反馈队列调度：允许进程在**队列之间移动**。根据**不同 CPU 区间**的特点以区分进程。如果进程使用过多 CPU 时间会被转移到更低优先级队列。在较低优先级队列中等待时间过长的进程会被转移到更高优先级队列。这种形式的老化可以阻止饥饿。最**通用**的 CPU 调度算法，可被配置以**适应**系统设计，但是**最复杂**。

8. 根据进程的到达和执行时间，画出相应算法的甘特图，并求出平均等待时间。

进程	到达时间	执行时间
P1	0	3
P2	1	2
P3	2	3
P4	3	6

（1）抢占式最短作业优先调度

（2）轮转法调度（时间片大小为 2）

（3）高响应比

9. 以下是四个进程的到达时间和运行时间。

	到达时间	运行时间
P1	0	12
P2	1	8
P3	2	3
P4	3	6

分别画出 FIFO 和 SJF 调度的甘特图，并计算平均等待时间。

10. 临界区设计的基本要求；信号量是怎么设计来满足这些要求的？

（1） **互斥：忙则等待。**进程不同时在临界区内执行。

（2） **前进：有空让进。**当无进程在临界区执行时，若有进程进入应允许。

（3） **有限等待**：进程进入临界区的要求必须在有限时间内得到满足。

（4） **让权等待**：等待的时候可以选择释放 CPU 执行权（非必须）。

临界区是进程中访问临界资源的代码段。不放非必要的代码（并发度降低）。实现进程互斥：分析问题，确定临界区；设置互斥信号量，初值为 1；临界区之前对信号量进行 P 操作；临界区之后对信号量执行 V 操作。

11. 写出下面程序的输出结果，并解释这样输出的原因。

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int a = 0;
int main() {
        pid_t pid = fork();
        if (pid == 0) {
                a = 2;
                printf("child leaving\n");
        } else {
                wait(NULL);
                printf("a=%d\n", a);
```

```
        }
        return 0;
}
```

child leaving

a=0

程序 fork()了一次，产生一个子进程。父进程和子进程并行运行，直到父进程执行 wait(NULL)，即 wait(0)。wait(0)表示父进程会被阻塞，直到子进程的状态发生变化，即从运行态到终止态，才会被唤醒。所以先输出子进程运行结果，后输出父进程运行结果。由于子进程执行 a=2 时发生写时复制，父子进程有独立的数据段，父进程输出 0 不变。

12. 两个进程 T1 和 T2 并发执行，共享变量 x，初值为 1，T1 使 x+1，T2 使 x-1，过程如下。问两个进程结束后 x 有多少种可能取值？有哪些方法使结果唯一？选取一种方法修改下面的程序，保证两进程结束后结果唯一。

T1	T2	
Load R1,x Inc R1 Store x,R1	Load R2,x Dec R2 Store x,R2	将 x 取到寄存器中 将寄存器的值放回 x

0, 1

13. 简述阻塞、饥饿、死锁、死循环的区别。

阻塞是进程正在等待某一事件而暂停运行，由运行态变成阻塞态，是进程自身的一种**主动**行为。处于阻塞态的进程可能发生死锁或饥饿，也可顺利向前推进。

饥饿、死锁和死循环都是进程**无法顺利向前推进**的现象（故意设计的死循环除外）。

死锁一定是“循环等待对方手里的资源”导致的，因此如果有死锁现象，那**至少有两个**进程同时发生死锁。另外，死锁的进程一定处于**阻塞态**。

可能**只有一个**进程发生**饥饿**。发生饥饿的进程既可能是**阻塞态**（如长期得不到需要的 I/O 设备），也可能是**就绪态**（长期得不到处理机）。

可能**只有一个**进程发生死循环。**死循环**的进程可以上处理机运行（可以是**运行态**），只不过无法像期待的那样顺利推进。

死锁和饥饿问题是由于**操作系统分配资源不合理**导致的，而**死循环**是由**代码逻辑的错误**导致的。

死锁和饥饿是**管理者**（操作系统）的问题，**死循环**是**被管理者**的问题。

14. 产生死锁的必要条件。

互斥条件：只有对必须互斥使用的资源的争抢才会导致死锁。将**临界资源改造为可共享**使用的资源（可行性不高）。

不剥夺条件：进程所获得得资源在未使用完之前，不能由其他进程强行夺走，只能主动释放。**让权；剥夺**（考虑优先级）。（实现复杂；剥夺可能导致部分工作实效；反复导致系统开销大；可饥饿）

请求和保持条件：进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源又被其他进程占有，此时请求进程被阻塞，但又对自己已有的资源保持不放。**静态分配**（资源利用率低；可饥饿）。

循环等待条件：存在一种进程资源的循环等待链，链中的每一个进程已获得的资源同时被下一个进程所请求。**顺序资源分配法**（不方便增加新的设备；顺序不一致资源浪费；编程麻烦）。

15. CPU 是进程运行必需的资源，为什么进程不会因等待 CPU 而发生死锁？

不满足**不剥夺条件**。对可剥夺的资源（CPU）的竞争是不会引起死锁的。

16. 简述银行家算法避免死锁的过程（包括变量定义、安全判别算法等）。

设系统中共有 **n 个进程**和 **m 种资源**类型。

（1） **安全性算法**：确定计算机系统是否处于安全状态

①向量 **finish[n]**存储进程是否已经完成，初始状态为 **false**；向量 **work[m]**存储当前每种资源的剩余可用量，初始值为**资源总量**。

②寻找是否存在 **finish=false** 且**所有**资源 **need<=work** 的进程，如果存在，让这个进程获得所需要的资源执行结束，然后释放资源，令它的 **finish=true**，总的 **work 加**上该进程原来占有的资源。

③**循环**执行②直到没有符和条件的进程。此时若 **finish 全部为 true**，则系统处于安全状态，能获得一个安全序列。

（2） **资源请求算法**：判断是否可安全允许请求

初始状态：allocation[n][m]，need[n][m]，available[m]

①确定 **request<=need** 否则**出错**；**request<=available**，否则**等待**；

②按照需求**假分配**，**修改**系统当前的 **available、need**，然后根据**安全性算法**判断假分配以后系统状态是否**安全**。如果安全，该分配得到**允许**。

17. 简述死锁预防、死锁避免并比较区别。

死锁预防是一组方法，需要确定**至少一个必要条件不成立**。

死锁避免要求操作系统事先得到有关进程**申请使用资源**的额外信息。当进程申请资源时，若发现满足该资源的请求可能导致死锁发生，则拒绝该申请。

两种方法都不允许死锁发生。**动态**的**死锁避免**会因为追踪当前资源分配成本增加运行**成本**，但是相对于**静态**的**死锁预防**方法它允许更多的**并发**使用资源，所以**系统吞吐量**大于死锁预防。

18. 使用银行家算法判断是否有死锁？如果有死锁，那么哪些进程死锁？

$request_1(a,b,c) < need_1(a',b',c')$

$request_1(a,b,c) <available(a'',b'',c'')$

假分配，则 available 变为(p,q,r)

系统状态序列见下表：

process\resource	allocation			need			available		
	A	B	C	A	B	C	A	B	C
P_0									
P_1									
P_2									
P_3									

进行安全性分析，见下表：

process\resource	work			allocation			need			work+allocation		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0												
P_2												
P_1												
P_3												

根据**安全性算法**求出一个**安全序列<P_0,P_2,P_1,P_3>**，系统仍处于**安全状态**。请求被**允许**。

19. 使用死锁检测算法判断是否有死锁？如果有死锁，那么哪些进程死锁？

	allocation	need	available
p1	0，1，0	0，1，0	0，0，0
p2	2，0，0	1，0，0	
p3	0，0，3	0，0，0	
p4	2，1，1	2，1，0	
p5	0，0，2	0，0，2	

process\resource	work			allocation			need			work+allocation		
	A	B	C	A	B	C	A	B	C	A	B	C
P_3	0	0	0	0	0	3	0	0	0	0	0	3
P_5	0	0	3	0	0	2	0	0	2	0	0	5

满足进程 P_3 和 P_5 的需求后，**work 为**(0,0,5)，此时不能满足余下任一进程的需求，系统出现死锁，因此当前系统处在**非安全状态**。P_1 、P_4 和 P_4 死锁。

20. 页大小为 1024 字节，计算逻辑地址 2560 和 4220 对应的物理地址

页号	块号
0	20
1	30
2	60
3	10

2560=1024*2+512

4220=1024*4+124

60*1024+512=61952

页号 4 越界。逻辑地址 4220 非法。

21.

（1）

一个进程的页表见下面的表格，页的大小为1024字节。为执行指令 MOV AX, [2560]和 MOV BX, [4098]，计算逻辑地址2560和4098对应的物理地址。

页号	帧号
0	20
1	30
2	18
3	80

注：本题中所有数字均为10进制。[2560]表示访问逻辑地址为2560的存储器，AX及BX均为寄存器名，MOV为移动数据的汇编指令

2560=1024*2+512

4098=1024*4+3

18*1024+512=18944

页号 4 越界。逻辑地址 4098 非法。

（2）单级页表；TLB 命中率为 90%，访问 TLB 需 5ns，访问主存为 25ns，求有效内存的访问时间。

0.9*(5+25)+0.1*(5+25+25)=32.5ns

22. 在某个分页管理系统中，某一个作业有 4 个页面，被分别装入到主存的第 3、4、6、8 块中，假定页面和块大小均为 1024 字节，当作业在 CPU 上运行时，执行到其地址空间第 500 号处遇到一条传送命令： MOV 2100,3100 画地址转换图计算出 MOV 指令中两个操作数的物理地址。
在页表中，逻辑页(0,1,2,3)对应物理块(3,4,6,8)，页面大小 L 为 1024 字节。 逻辑地址 A1=2100，页号 P1=(int)(2100/1024)=2，页内偏移量 W1=2100-2×1024=52，对应的物理块号为 6，则 A1 对应的物理地址 E1=6×1024+52=6196。 逻辑地址 A2=3100，页号 P2=(int)(3100/1024)=3，页内偏移量 W2=3100-3×1024=28，对应的物理块号为 8，则 A2 对应的物理地址 E2=8×1024+28=8220。
23. **请求**分页系统中，页表项中包含哪些数据项？它们的作用？
a.在请求分页系统中，其页表项中包含的数据项有**页号、内存块号、状态位 P、访问字段 A、修改位 M 和外存地址**；
b.其中**状态位 P** 指示该页是否调入内存，供程序访问时参考；
c.**访问字段 A** 用于记录本页在一段时间内被访问的次数，或最近已有多长时间未被访问，提供给**置换算法**选择换出页面时参考；
d.**修改位 M** 表示该页在调入内存后是否被修改过；
e.**外存地址**用于指出该页在外存上的地址，通常是物理块号，供调入该页时使用。
24. 画出分页内存管理方案的过程图，描述流程、过程中硬件或软件所起的作用。
25. 分页式文件系统，页表的主要数据结构，要使用这个分页结构需要哪些硬件支持。
CPU-地址转换机构-越界检查机构-
页表（慢表）：丨页号丨页内偏移量丨
相联存储器 TLB（快表）：丨页号丨页内偏移量丨
页表寄存器 PTR：丨页表起始地址丨页表长度丨
26. 当前页表如下。页大小为 1024 字节，该程序分配 2 个帧，页号 0 先装入内存。采用先进先出和局部置换策略，现在访问逻辑地址为 3000 的字节，问在这个过程中发生了什么主要事件并写出置换后的页表。

页号	帧号	Valid/Invailid
0	130	Valid
1	570	Valid
2	-1	Invalid
3	-1	Invalid

3000=1024*2+952，页号为 2<4，状态位为 Invalid，发生缺页中断，由于 2 个帧已满，采用 FIFO 算法和局部置换策略，且页号 0 先装入内存，置换出，状态位置为 Invalid，将页号 2 对应的帧号修改为 130，状态位置为 Valid。最后，访问物理地址为 130*1024+952=134072 的字节。
置换后的页表为：

页号	帧号	Valid/Invalid
0	130	Invalid
1	570	Valid
2	130	Valid
3	-1	Invalid

27. 在一个请求分页系统中，用 FIFO，LRU 写出下面的置换过程 4,3,2,1,4,3,5,4,3,2,1,5，当分配给该作业的物理块数 M 分别为 3、4 时分别计算缺页次数和缺页率，并说明 LRU 优于 FIFO 的原因。

M=3：

	页面访问	4	3	2	1	4	3	5	4	3	2	1	5
FIFO	物理块 1	4	4	4	1	1	1	5			5	5	
	物理块 2		3	3	3	4	4	4			2	2	
	物理块 3			2	2	2	3	3			3	1	
	是否缺页	✓	✓	✓	✓	✓	✓	✓			✓	✓	
LRU	物理块 1	4	4	4	1	1	1	5			2	2	2
	物理块 2		3	3	3	4	4	4			4	1	1
	物理块 3			2	2	2	3	3			3	3	5
	是否缺页	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓

FIFO：缺页次数=9，缺页率=75%

LRU：缺页次数=10，缺页率=83.3%

M=4：

	页面访问	4	3	2	1	4	3	5	4	3	2	1	5
FIFO	物理块 1	4	4	4	4			5	5	5	5	1	1
	物理块 2		3	3	3			3	4	4	4	4	5
	物理块 3			2	2			2	2	3	3	3	3
	物理块 4				1			1	1	1	2	2	2
	是否缺页	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓
LRU	物理块 1	4	4	4	4			4			4	4	5
	物理块 2		3	3	3			5			5	1	1
	物理块 3			2	2			3			3	3	3
	物理块 4				1			1			2	2	2
	是否缺页	✓	✓	✓	✓			✓			✓	✓	✓

FIFO：缺页次数=10，缺页率=83.3%

LRU：缺页次数=8，缺页率=66.7%

FIFO 优先淘汰最先进入内存的页面，但是可能置换出去的是**很久之前初始化并且一直在用**的变量。性能很差，存在 **Belady 异常**。上述结果可以看出，对 FIFO 而言，增加分配给作业的内存块数反而出现缺页次数增加的异常现象。给进程分配 4 个帧产生的页面错误率比分配 3 个帧还多。

LRU 优先淘汰最近最久没访问的页面。和最优置换（OPT）一样，都**没有 Belady 异常**（它们属于同一种算法，叫做**栈算法**）。效率较高，是一种经常被使用的页置换算法。

28. FIFO 和 LRU 的页面置换算法哪个更好，为什么？

同 26

29. 增强的二次机会算法的基本思想；为什么会发生抖动，怎么解决？

若用（访问位，修改位）的形式表述，则

第一轮：淘汰（0,0）

第二轮：淘汰（0,1），并将扫描过的页面访问位都置为 0

第三轮：淘汰（0,0）

第四轮：淘汰（0,1）

产生抖动的主要原因是进程频繁访问的页面数目高于可用的物理块数（分配给进程的物理块不够）。

工作集指在某段**时间间隔**里，进程**实际访问**页面的集合。**驻留集**（请求分页存储管理中给进程分配的内存块的集合）大小不小于工作集大小。可变分配？

30. 什么是颠簸？系统提供哪几种方法避免颠簸？如果发生颠簸该怎么处理？什么是工作集模型？

进程没有“足够”的帧，需要进行页调度，而调度出去的页再次被需要，所以又很快被调回来，导致进程用于页调度的时间多于执行的时间。

31. 图示说明共享页面，共享代码和共享数据的限制
32. 三种磁盘空间分配方法（contiguous、linked、indexed）的优缺点。

①**连续分配**：实现**简单**，**随机**存取速度快、效率高，适合文件内容**不进行变动**的情况（对换区的分配方式）。但难以进行**文件扩展**，需要**提前声明**文件大小。产生**外**碎片。

②**链接分配**：实现**简单**，只需要**首**地址。没有**外**碎片问题。文件可以**扩展**，不需要提前声明文件大小。但每个文件块都有指针，占用**空间**。无法实现**随机**读取。**可靠性差**，一个中间数据块中指针的丢失都会导致链的断裂。（文件配置表 FAT 改进）

③**索引分配**：**随机**访问。可扩展，文件不需要提前声明大小。没有外碎片。但浪费**空间**，如一个文件只有两块大小，采用链接分配只浪费一个指针空间，采用索引分配浪费 1 块索引块的空间。（索引块>1：链接/多层索引/组合：一层两层三层索引）

33. 基于磁盘的分配方法，设计一种高效的磁盘**空闲**空间分配和回收方法，说出基本思想，操作过程和数据结构。

①空闲表法

②空闲链表法：[空闲盘块链]将所有空闲磁盘块用链表链接起来，并将指向第一个空闲块的头指针保存在磁盘的超级块上，同时也缓存在内存中。[空闲盘区链/计数 counting]类似于内存动态分区。

③位示图法：将空闲空间表现为位图，每块空闲空间用一位表示，如果空闲则为 1，已分配则为 0。

④成组链接法：把**顺序的 n 个空闲扇区地址**保存在**第一个空闲扇区**内，其**最后一个空闲扇区**内则保存**另一组顺序空闲扇区的地址**，如此继续，直至所有空闲扇区均予以链接。系统只需要保存一个指向**第一个空闲扇区的指针**。

34. 某磁盘文件区 16GB，每个磁盘块为 1KB，

（1）若空闲存储空间采用位图管理方法那么位图需占用多少个盘块

（2）若采用 FAT，FAT 中每个盘块地址用 4 字节表示，问 FAT 表需要几个盘块

（1）$16*2^{34}/2^{10}/8/2^{10}=2^{11}$。

（2）$16*2^{34}/2^{10}*4/2^{10}=2^{16}$

35.

某磁盘逻辑地址 32 位，页大小 16K，页表项大小 4B

1、采用多层页表结构，该采用几层页表？页偏移多少比特？画出地址分配。

2、对逻辑地址 54321（10 进制），简述求实际地址的过程（忽略缺页中断）

3、CPU 和操作系统在分页中各自承担了那些工作，简要说明

36. 一级目录下，文件写入不可更改，可不断新建，

（1）问连续、链接、索引哪个比较符合这个文件系统，为啥？其 FCB 中需要存储什么之类的

（2）FCB 连续存储还是跟着文件存储比较好，为啥？

37.

（1）如何通过文件名找到 FCB

39

（2）三种磁盘空间分配方法（连续、链接、索引）FCB 中存的内容

①连续分配：file start length

②链接分配：file start end

③索引分配：file index_block

38. 文件目录为一级目录，文件能一次性写入，且写入后无修改，但是可以创建多个新文件。

（1）为文件分配磁盘空间的方法有连续、链接、索引，用哪种方法(连续)比较好？解释原因（连续的优点）。FCB 包含什么内容？

实现**简单**，**随机**存取速度快、效率高，适合文件内容**不进行变动**的情况（对换区的分配方式）。

file start length

FCB 是种数据结构，包含了一个文件的所有结构信息，通过目录表里的叶结点中文件名和 FCB 号的一一映射，我们可以根据文件名获取该文件的 FCB。FCB 包含文件权限，文件日期（创建、访问、修改），文件拥有者、分组、ACL，文件大小，文件数据块或指向文件数据块物理地址的指针。

（2）FCB 是集中存储比较好还是与每个相关的文件一起存储比较好？解释原因。(集中存储)

39. 简要描述操作系统中打开文件的工作过程。

用户给 **open()**传入一个文件的**逻辑路径名**，这时先将该文件系统的**目录结构**加载进内存，根据文件名，操作系统会首先对**系统范围内的打开文件表**进行**搜索**（节省时间）。

如果该文件已经被其他进程打开了，则直接将该**进程的打开文件表中的指针**指向系统范围打开文件表的这一项，同时，系统打开文件表该文件**引用计数**加 1。

如果该文件在系统范围文件表中**不存在**，说明该文件第一次打开，则对该文件系统的目录表进行搜索，依次查找到叶结点，叶结点包含了一个该**文件控制节点号**，即控制节点的物理位置指针，将这个指针返回给用户，同时在系统范围打开文件表中新注册一行这个文件的信息，将该进程的打开文件表中指针指向这条新信息，open()操作的任务就完成了。

40. 为什么访问文件需要先 open()？以及该操作之后，操作系统产生了什么结构？

进程打开文件表：读写指针、访问权限。

41. 基于文件的目录结构，写出实现文件共享的几种方法。

①基于**索引结点**的共享方式（硬链接）。各个用户的目录项指向同一个索引结点，索引结点中有一个链接计数器 count。某用户想删除文件时，指删除该用户的目录项，且 count--，只有 count==0 时才能真正删除文件数据和索引结点，否则会导致指针悬空。

②基于**符号链**的共享方式（软连接）。在一个 Link 型的文件中记录共享文件的存放路径，操作系统根据路径找到文件一层层查找目录，最终找到共享文件。即使软连接指向的共享文件已被删除，Link 型文件依然存在，只是通过 Link 型文件中的路径去查找共享文件会失败。由于软链接会有多次磁盘 I/O，所以速度更慢。

42. 某磁盘有 5000 个磁道，从 0-4999。当前磁头的位置为 143，前一个请求的位置为 144。给出当前的请求队列 83、252、143、…、1774（题中共给了 9 个），分别用下面的算法求出寻道路径长度。

（1）SSTF（最短寻找时间优先，贪心）

（2）SCAN（扫描/电梯，解决饥饿问题）

（3）LOOK（提前结束）

（4）C-SCAN（循环扫描，解决 SCAN 对各个位置磁道的响应频率不平均）

（5）C-LOOK（提前结束）

43. FCFS 在 CPU 调度、磁盘调度、页置换中分别的优缺点。

①

②

③**磁盘调度**：算法比较公平，但是因为磁臂摆动幅度大导致平均寻道距离大，效率低。

44. 非阻塞和阻塞 I/O 是什么，主要有什么不同，分别用在哪里？

阻塞 I/O，当应用程序发出一个**阻塞系统调用**时，应用程序挂起，应用程序从运行队列转入等待队列。等系统调用完成之后再回到就绪队列，在合适的时候继续运行。**绝大多数操作系统**为应用程序提供的都是阻塞系统调用，因为它代码更加简单，更容易理解。

非阻塞 I/O，一个**非阻塞调用**在较长时间内不中止线程执行，它会很快返回，其返回值表示已经传输了多少字节。在**多个程序协作完成 I/O 的工作**时，可能会使用非阻塞 I/O。

45. 访存操作可能会导致 IO 的进行，某进程读写文件时并没有 IO 设备执行，为什么？

假脱机？

46. 缓冲机制的三种用途并举例。

①处理数据流中**生产者与消费者之间的速度差异**。使用**双缓冲机制**，一个缓冲接收生产者数据，另一个缓冲写给消费者，若生产者较慢，那么当第一个缓冲写满了以后两个缓冲调换，完成了生产者与消费者的分离解耦。

②协调**传输数据大小不一致**的设备。如**网络传输**中发送端将大消息分成若干网络包，接收端将它们放在**重组缓冲区**内。

③**支持应用程序 I/O 的复制语义**。当用作 **buffer 的页面被修改**后，复制出一个新的页面，原来的页面用于 write()，新的页面用于修改。

47. 具体解释 bad-section mapping、prefetching、buffer、caching 的概念和用途。

缓冲区（buffer）是用来保存两个设备之间或在设备和应用程序之间所传输数据的内存区域。

48. 为何没有“进程安全”？

不会

49. 结合所学的计算机知识，简述保护的概念（从硬件层面、文件管理、设备管理、存储管理、进程同步等方面）。

文件保护：口令保护、加密保护、访问控制（用一个访问控制表 ACL 记录各个用户（或各组用户）对文件的访问权限）。

设备保护：在 UNIX 系统中，设备被看做是一种特殊的文件，每个设备也会有对应的 FCB.当用户请求访问某个设备时，系统根据 FCB 中记录的信息来判断该用户是否有相应的访问权限，以此实现“设备保护”的功能。

50. 为保证可靠性，操作系统提供了那些措施，请从进程同步、文件管理、存储管理、设别管理等方面分析。

PV 操作题目分析步骤：

a) 关系分析。找出题目中描述的各个进程，分析它们之间的同步、互斥关系。

b) 整理思路。根据各进程的操作流程确定 P、V 操作的大致顺序。

c) 设置信号量。设置需要的信号量，并根据题目条件确定信号量的初值。（互斥信号量初值一般为 1，同步信号量初始值要看对应资源的初始值是多少。

d) 实现互斥的 P 操作一定要在实现同步的 P 操作之后（否则会发生死锁）。两个 V 操作顺序可以交换。

51.

七、从前有座山，山上有座庙，山下有口井。庙里小和尚需要挑水。有人舞担，有人拿桶，有人诵挑水秘诀。挑水时，三个和尚必须一人持担，一人拿桶、一人诵挑水秘诀(同时进行)后方能挑水。每个和尚都是先喜欢诵诀，其次持担、其次持桶。请写出信号量和相关伪代码。

```
int action=0
semaphore finish1=0, finish2=0, finish3=0
semaphore action_mutex=1
CoBegin
Monk1() {
    P(action_mutex)
    if(action==2) {
        V(action_mutex)
        V(finish2)
        V(finish3)
    } else {
        action++
        V(action_mutex)
        P(finish1)
    }
    挑水
}
Monk2() {
    P(action_mutex)
    if(action==2) {
        V(action_mutex)
        V(finish1)
        V(finish3)
    } else {
        action++
        V(action_mutex)
        P(finish2)
    }
    挑水
}
Monk3() {
    P(action_mutex)
    if(action==2) {
        V(action_mutex)
        V(finish1)
        V(finish2)
    } else {
        action++
        V(action_mutex)
        P(finish3)
    }
    挑水
}
CoEnd
```

52. 两人拿棋子，一次一个，不相差 M 个，写出并发的伪代码。

```
semaphore mutex=1
semaphore a=M-1, b=M-1
CoBegin
A() {
    while(1) {
        P(a)
        P(mutex)
        get-pawn
        V(mutex)
        V(b)
    }
}
B() {
    while(1) {
        P(b)
        P(mutex)
        get-pawn
        V(mutex)
        V(a)
    }
}
CoEnd
```

53. 一个场地最多容纳 22 人，其中有打篮球的（不妨设为活动 A），打羽毛球的（不妨设为活动 B）。

（1）如果人数达到上限则等待。

（2）如果活动 A 的人数比活动 B 的人数多 5 人，则参与 A 的同学需等待；如果活动 B 的人数比活动 A 的人数多 5 人，则参与活动 B 的同学需等待。（-6<A-B<6）

（3）活动 A 和活动 B 的同学可随时离开。

请用“参与 A”“参与 B”“离开 A”“离开 B”和信号量来说明这个过程。

```
semaphore capacity=22, a=5, b=5
CoBegin
A() {
    while(1) {
        P(a)
        P(capacity)
        参与 A
        V(b)
        离开 A
        V(a)
        V(capacity)
        P(b)
    }
B() {
    while(1) {
```

```
            P(b)
            P(capacity)
            参与 B
            V(a)
            离开 B
            V(b)
            V(capacity)
            P(a)
        }
    }
    CoEnd
```

54. 某银行提供 1 个服务窗口和 10 个供顾客等待的座位。顾客到达银行时，若有空座位，则到取号机上领取一个号，等待叫号。取号机每次仅允许一位顾客使用。当营业员空闲时，通过叫号选取一位顾客，并为其服务。

```
semaphore service=0, customers=0, seats=10
semaphore machine_mutex=1
CoBegin
Process Cashier {
    while(1) {
        P(customers)
        V(service)
        V(seats)
        serve
    }
}
Process Customer i {
    P(seats)
    P(machine_mutex)
    get-number
    V(machine_mutex)
    V(customers)
    P(service)
    accept-service
}
CoEnd
```

55. 现有 8 个协作的多任务（A,B,C,D,E,F,G,H）并发执行，他们满足下列条件：任务 A 必须领先于任务 B,C,E，任务 E 和 D 必须领先于任务 F，任务 B 和 C 必须领先于任务 D，而任 F 必须领先于任务 G 和 H，用信号量分别写出每个任务的程序。

```
semaphore ab=0, ac=0, ae=0, bd=0, cd=0, df=0, ef=0, fg=0, fh=0
CoBegin
A() {
    finish-A
    V(ab)
    V(ac)
```

```
        V(ae)
    }
    B() {
        P(ab)
        finish-B
        V(bd)
    }
    C() {
        P(ac)
        finish-C
        V(cd)
    }
    D() {
        P(bd)
        P(cd)
        finish-D
        V(df)
    }
    E() {
        P(ae)
        finish-E
        V(ef)
    }
    F() {
        P(ef)
        finish-F
        V(fg)
        V(fh)
    }
    G() {
        P(fg)
        finish-G
    }
    H() {
        P(fh)
        finish-H
    }
    CoEnd
```

56. 甲乙丙三人疫情结束后一起聚餐，吃完饭后，三人一起看电影。用伪代码写出三人一起行动的过程，并且要防止死锁的发生。

```
semaphore arrive_ab=0, arrive_ac=0, arrive_ba=0, arrive_bc=0, arrive_ca=0, arrive_cb=0
semaphore ab=0, ac=0, ba=0, bc=0, ca=0, cb=0
CoBegin
A() {
```

```
        V(arrive_ab)
        V(arrive_ac)
        P(arrive_ba)
        P(arrive_ca)
        eat
        V(ab)
        V(ac)
        P(ba)
        P(bc)
        watch-movie
    }
    B() {
        V(arrive_ba)
        V(arrive_bc)
        P(arrive_ab)
        P(arrive_cb)
        eat
        V(ba)
        V(bc)
        P(ab)
        P(cb)
        watch-movie
    }
    C() {
        V(arrive_ca)
        V(arrive_cb)
        P(arrive_ac)
        P(arrive_bc)
        eat
        V(ca)
        V(cb)
        P(ac)
        P(bc)
        watch-movie
    }
    CoEnd
```

概念：

API（application programming interface，应用编程接口）：应用程序开发人员根据应用程序接口来设计程序。API 为方便应用程序员规定了一组函数，包括每个函数的输入参数和返回值。P43

Multiprogramming（多道程序设计）：多道程序设计通过安排作业（编码与数据）使得 CPU 总有一个执行作业，从而提高 CPU 利用率。P13

Time-Sharing Operating System（分时操作系统）：分时系统（或多任务）是多道程序设计的自然延伸。P13

User mode（用户模式）：为了确保操作系统的正确运行，必须区分操作系统代码和用户代码的执行。至少需要两种单独运行模式：用户模式和内核模式。当计算机系统执行用户应用时，系统处于用户模式。P15

Kernel mode（内核模式）：当用户应用通过系统调用，请求操作系统服务时，系统必须从用户模式切换到内核模式，以满足请求。

Privileged instruction（特权指令）：将可能引起损害的机器指令作为特权指令，并且硬件只有在内核模式下才允许执行特权指令。P15

System call（系统调用）：事件发生通常通过硬件或软件的中断来通知。软件也可通过执行特别操作即系统调用，以触发中断。P5

Process（进程）：进程为执行程序。进程是现代分时操作系统的工作单元。P72

IPC（Interprocess communication, 进程间通信）：协作进程需要有一种进程间通信机制，以允许进程相互交换数据与信息。P84

Thread（线程）：每个线程是 CPU 使用的一个基本单元；它包括线程 ID、程序计数器、寄存器和堆栈。P112

User thread（用户线程）：有两种不同方法来提供线程支持：用户层的用户进程或内核层的内核进程。用户线程仅存在于用户空间之中。用户线程受内核支持，而无需内核管理。 P116

Kernel thread（内核线程）：内核线程由操作系统来直接支持与管理。

Preemptive scheduling（抢占调度）：需要进行 CPU 调度的情况可分为以下四种：

- 当一个进程从运行状态切换到等待状态时。
- 当一个进程从运行状态切换到就绪状态时。
- 当一个进程从等待状态切换到就绪状态时。
- 当一个进程终止时。

如果调度只能发生在第 1 种和第 4 种情况下，则调度方案称为非抢占的或协作的；否则，调度方案成为抢占的。P139

Atomic operation（原子操作）：一个完整的没有中断的操作，要么全做要么全不做，是一种不可分操作。这些原子操作在核心态下运行，常驻内存。

Primitive（原语）：一段完成一定功能的执行期间不能被中断的程序段。

Critical resource（临界资源）：系统中可以供给多个进程使用但是同一段时间内却只允许一个进程访问的资源。

Critical section（临界区）：进程中访问临界资源的代码段。

Race condition（竞争条件）：多个进程并发访问和操作同一数据，且执行结果和访问发生的特定顺序有关。

Protection domain（保护域）：保护域指定了进程可以访问的资源。每个域定义了：一组对象和可以对每个对象调用的操作类型。P418

Semaphore（信号量）：一个信号量 S 是个整型变量，它除了初始化外只能通过两个标准原子操作：wait()和 signal()来访问。

Monitor（管程）：一种重要的、高级的同步工具。P189 管程是一种特殊的软件模块，由共享数据结构、对数据结构初始化的语句和一组用来访问数据结构的过程（函数）组成。各外部进程/线程只能通过管程提供的“特定”入口才能访问共享数据，每次仅允许一个进程在管程内执行某个内部过程。

Deadlock（死锁）：当一个进程申请资源时，如果这是没有可用资源，那么这个进程进入等待状态。有时，如果所申请的资源被其他等待进程占有，那么该等待进程有可能再也无法改变状态。这种情况称为死锁。P212

Safe sequence（安全序列）：只有存在一个安全序列，系统才处于安全状态。进程序列<P1,P2,…,Pn>在当前分配状态下为安全序列是指：对于每一个 Pi，Pi 仍然可以申请的资源数小于当前可用资源加上所有进程 Pj（其中 j<i）所占用的资源。P219

Safe（安全状态）：如果系统能按一定顺序为每个进程分配资源（不超过它的最大需求），仍然避免死锁，那么系统的状态就是安全的。P219

FAT（File-allocation table, 文件分配表）：链接分配的一个重要变种是文件分配表的使用。这个简单而有效的磁盘空间分配方法用于 MS-DOS 操作系统。P346

FCB（File Control Block，文件控制块）：逻辑文件系统通过文件控制块来维护文件结构。文件控制块包含有关文件的信息，包括所有者、权限、文件内容的位置等。P338

File sharing（文件共享）：允许多个用户共享文件，将文件共享扩展多个文件系统。p327

Blocking I/O（阻塞 I/O）：当应用程序执行阻塞系统调用时，应用程序的执行就被挂起。P401

Block devices（块设备）：块设备以字节块为单位来传输。块设备接口为磁盘驱动器和其他基于块设备的访问，规定了所需的各个方面。P399

TLB（Translation lookaside buffer，转址旁路缓存/转换表缓冲区）：

File directory（文件目录）：p319（device directory/directory）？包含文件系统的的每个卷也应包含有关系统内的文件信息。这些信息保存在设备目录或卷目录表中。设备目录（更常称为目录）记录卷上的所有文件的信息，如名称、位置、大小和类型等。

Working directory：无

Logical formatting（逻辑格式化）：在使用磁盘存储文件之前，操作系统仍需要将自己的数据结构记录在磁盘上 。分为两步。第一步是将磁盘分为柱面组成的多个分区。第二步是逻辑格式化，或创建文件系统。操作系统将初始的文件系统数据结构存储到磁盘上。P375

RAID（redundant array of inexpensive/independent disks，磁盘冗余阵列）：多种磁盘组织技术统称为磁盘冗余技术，通常用于处理性能与可靠性问题。P378

SPOOLing（Simultaneous Peripheral Operations On-Line，假脱机）：即外部设备联机并行操作，是利用进程与磁盘对脱机输入输出工作过程的模拟，因此又称假脱机。这种技术将慢速的字符设备（独占设备）虚拟成共享的虚拟设备，从而提高设备利用率和进程的推进速度。P405

Virtual machine（虚拟机）：虚拟机的核心思想就是将一套硬件设备抽象成多套。采用 CPU 调度和虚拟内存的技术，制造每一个进程都有自己单独的处理器和内存的假象。

VFS（Virtual file system / Virtual filesystem switch， 虚拟文件系统）：数据结构和程序用于隔离基本系统调用的功能与实现细节。因此，文件系统的实现由三个主要层构成。第一层为文件系统接口。第二层为虚拟文件系统。VFS 层提供两个重要功能：通过定义清晰的 VFS 接口，它将文件系统的通用操作和实现分开。它提供了一种机制，以唯一表示网络上的文件。第三层略。p342

Thrashing（抖动/颠簸）：如果进程没有需要支持活动使用页面的帧数，那么它会很快产生缺页错误。此时，必须置换某个页面。由于它的页面都在使用中，它会再次快速产生缺页错误，再一次置换必须立即返回的页面。这种高度的页面活动称为抖动。如果一个进程的调页时间多于它的执行时间，那么这个进程就在抖动。P283

Working set（工作集）：最近页面引用的页面集合称为工作集。如果一个页面处于活动使用状态，那么它在工作集中。工作集是程序局部的近似。P285

Device driver（设备驱动程序）：通常，操作系统为每个设备控制器提供一个设备驱动程序。该设备驱动程序负责设备控制器，并且为操作系统的其他部分提供统一的设备访问接口。

PV 例题

1. 生产者-消费者

```
semaphore empty=N, full=0
semaphore mutex=1
Producer() {
    while(1) {
        produce
        P(empty)
        P(mutex)
        add-to-buffer
        V(mutex)
        V(full)
    }
}
Consumer() {
    while(1) {
        P(full)
        P(mutex)
        remove-from-buffer
        V(mutex)
        V(empty)
        consume
    }
}
```

2. 有一个仓库，可以存放 A 与 B 两种产品，仓库的存储空间足够大，但要求:

（1） 每次只能存入一种产品（A 或 B）;

（2） 2-N＜A 产品数量 － B 产品数量 ＜M;

其中，N 和 M 是正整数。试用“存放 A”和“存放 B”和 wait、signal 描述产品 A 与产品 B 的入库过程。

```
semaphore a=M-1, b=N-1
semaphore mutex
A() {
    while(1) {
        P(a)
        P(mutex)
        deposit-A
        V(mutex)
        V(b)
    }
}
B() {
    while(1) {
        P(b)
        P(mutex)
        deposit-B
        V(mutex)
```

```
        V(a)
    }
}
```

3. 桌子上有一个盘子，每次只能向其中放入一个水果。爸爸专向盘子中放苹果，妈妈专向盘子中放橘子，儿子专等吃盘子中的橘子，女儿专等吃盘子中的苹果。只有盘子为空时，爸爸或妈妈才可向盘子中放一个水果;仅当盘子中有自己需要的水果时，儿子或女儿可以从盘子中取出。

4. 某寺庙有小和尚、老和尚若干。有一水缸，由小和尚用水桶从井中提水入缸，老和尚用水桶从缸里取水饮用。水缸可容 10 桶水，水取自同一井中。水井径窄，每次只能容一个水桶取水。水桶总数为 3 个。每次入、取缸水仅为 1 桶，且不可以同时进行。试用 P、V 操作给出小和尚、老和尚动作的算法描述。

```
semaphore buckets=3, empy=10, full=0
semaphore tank_mutex=1, well_mutex=1
OldMonk() {
    while(1) {
        P(full)
        P(buckets)
        P(tank_mutex)
        get-water
        V(tank_mutex)
        V(empty)
        drink-water
        V(buckets)
    }
}
Young_Monk() {
    while(1){
        P(empty)
        P(buckets)
        P(well_mutex)
        get-water
        V(well_mutex)
        P(tank_mutex)
        inject-water
        V(tank_mutex)
        V(buckets)
        V(full)
    }
}
```

5. 读者-写者

6. 哲学家进餐

7. 吸烟