1. **操作系统**是一组控制和管理计算机硬件和软件资源、合理地组织计算机工作流程以及方便用户的程序的集合。

2. **作业**是用户在一次解题或一个事务处理过程中要求计算机系统所做工作的集合，包括用户程序、所需的数据及命令等。

3. **并发性(concurrency)**是指两个或多个事件在**同一时间间隔**内发生；**并行性(parallelism)**是指两个或多个事件在**同一时刻**发生。

4. 操作系统的四大功能是 **处理机管理**、**存储器管理**、**设备管理**、**文件管理**。

5. 现代操作系统的两个最基本特征是 **并发** 和 **共享**。

6. 在操作系统中**引入进程**的目的**是为了使多道程序并发执行，以改善资源利用率及提高系统吞吐量**；而在操作系统中再**引入线程**，则**是为了减少程序并发执行时所付出的时空开销，使操作系统具有更好的并发性**。

7. **进程与线程的主要区别：**

1) **调度方面：**在传统的操作系统中，拥有资源和独立调度的基本单位都是进程；而在引入线程的操作系统中，线程是独立调度的基本单位，进程是资源拥有的基本单位。在同一进程中，线程的切换不会引起进程切换；在不同进程中，进行线程切换将会引起进程切换。
2) **拥有资源：**不论是传统操作系统还是设有线程的操作系统，进程都是拥有资源的基本单位；而线程不拥有系统资源（只有一点运行时必不可少的资源），但线程可以访问其所属进程的资源。
3) **并发性：**在引入线程的操作系统中，不仅进程之间可以并发执行，而且同一进程内的多个线程之间也可以并发执行。
4) **系统开销：**由于创建进程或撤销进程时，系统都要为之分配或回收资源，操作系统所付出的开销远大于创建或撤销线程时的开销。在进行进程切换时，涉及到整个当前进程 CPU 环境的保存及新调度到进程 CPU 环境的设置；而线程切换时，只需保存和设置少量寄存器内容，因此开销很小。另外，由于同一进程内的多个线程共享进程的地址空间，因此，这些线程之间的同步与通信非常容易实现，甚至无需操作系统的干预。

8. **临界资源：**一次仅允许一个进程使用的资源称为临界资源。

9. **进程同步**是指多个相互合作的进程，在一些关键点上可能需要互相等待或互相交换信息，这种相互制约关系称为进程同步。

10. 信号量及 P、V 操作（对进程的管理和控制的功能是通过执行各种原语来实现的。）：

信号量是一个确定的二元组(S,Q)，其中 S 是一个具有非负初值的**整型变量**，Q 是一个**初始状态为空的队列**。整型变量 S 表示系统中某类资源的数目，当其值小于 0 时，其绝对值表示系统中因请求该类资源而被阻塞的进程数目。除信号量的初值外，信号量的值仅能由 P 操作（又称为 Wait 操作）和 V 操作（又称为 Signal 操作）改变。

一个信号量的建立必须经过说明，即应该准确说明 S 的意义和初值（注意：**这个初值不是一个负值**）。每个信号量都有相应的一个队列，在建立信号量时，队列为空。

P、V 操作以**原语**方式实现，**信号量的值仅能由这两条原语加以改变。**P、V 操作的定义如下：

（1）P 操作。P 操作记为 P(S)，其中 S 为一个信号量，它执行时主要完成下述动作：

- S = S - 1;
- 若 S >= 0 则进程继续运行；否则（即 S < 0）阻塞该进程，并将它插入该信号量的等待队列中。

（2）V 操作。V 操作记为 V(S)，S 为一个信号量，它执行时主要完成下述动作：

- S = S + 1;
- 若 S > 0 则进程继续运行；否则（即 S <= 0）从信号量等待队列中移出第一个进程，使其变为就绪状态并插入就绪队列，然后再返回原进程继续执行。

11. 无论在生产者进程还是在消费者进程中，**P 操作的次序都不能颠倒**，否则将可能造成死锁。生产者-消费者问题的同步描述如下：

```
semaphore full = 0; /* 满缓冲单元的数目 */
semaphore empty = n; /* 空缓冲单元的数目 */
semaphore mutex = 1; /* 对有界缓冲区进行操作的互斥信号量 mutex 中文含义：互斥*/
main()
{
    cobegin
        producer();
        consumer();
    coend
}
producer()
{
    while(true)
    {
        生产一个产品；
        p(empty);
        p(mutex);
        将一个产品送入有界缓冲区；
        v(mutex);
        v(full);
    }
}
consumer()
{
    while(true)
    {
        p(full);
        p(mutex);
        从缓冲区中取出一个产品；
        v(mutex);
        v(empty);
        消费一个产品；
    }
}
```

如果将 p(full)和 p(mutex)位置互换，则可能导致死锁。这是因为对调 p(full)和 p(mutex)后，有可能出现这样一种特殊情况：某时刻缓冲区中没有产品且缓冲区中无进程工作（这时 full = 0，empty = n，mutex = 1），若系统此时调度消费者进程运行，它执行 p(mutex)并顺利通过（这时 mutex = 0），随后它执行 p(full)时因缓冲区中没有产品而受阻等待，等待生产

者进程生产产品并将产品放入缓冲区中；生产者进程生产产品后，执行 p(empty)并顺利通过，再执行 p(mutex)时，因缓冲区被消费者进程占据而无法进入。这样就形成了消费者进程在占有缓冲区资源的情况下，等待生产者进程放入产品，而生产者进程又无法进入缓冲区存放产品的僵局，此时这两个进程陷入死锁。随着进程的运行，其他生产者及消费者也会陷入死锁状态。

V 操作的功能是释放资源并唤醒等待资源的进程，因此只将 v(mutex)和 v(empty)位置互换对系统基本没有影响。由于所有生产者及消费者需互斥使用缓冲区资源，因此先执行 v(mutex)可能会使整个系统效率更高（因为总是先释放资源）。

12. 当调度程序为某就绪状态的进程分配了处理机时，该进程便由就绪状态变为执行状态；正在执行的进程因等待发生某事件而无法执行时，该进程由执行状态变为等待状态；当处于等待状态的进程所等待的事件发生时，该进程由等待状态变为就绪状态；正在执行的进程如因时间片用完而暂停执行，该进程由执行状态转变为就绪状态。

13. 在引入线程的操作系统中，**资源分配**的基本单位是**进程**，**CPU 分配**的基本单位是**线程**。用户级线程只存在于用户级，操作系统内核不知道用户级线程的存在，其调度及管理以进程为单位；内核级线程的创建、撤销及调度都由操作系统内核来实现。

14. 进程的基本特性是**动态性**、**并发性**、**独立性**、**异步性**及**结构特征**。

15. 一个作业从提交开始直到完成往往要经历下述三级调度：作业调度（宏观调度、高级调度或长程调度）、进程调度（微观调度、低级调度或短程调度）、交换调度（中级调度或中程调度）。

16. 进程调度方式分为**剥夺方式**和**非剥夺方式**。

17. 进程调度算法：**先来先服务调度算法**、**最高优先权优先调度算法**、**时间片轮转调度算法**、**多级反馈队列调度算法**（设置多个就绪队列，第一个队列优先权最高，其余队列的优先权逐次降低。进程所在队列的优先权愈高，其相应时间片愈短。在每个队列按先来先服务的原则排队等待调度，直到最后一个队列中使用时间片轮转调度算法。当处理机正为第 i 个队列中的某进程服务时，若又有新进程进入优先权较高的队列，新进程将抢占正在运行进程的处理机）。

18. 死锁

产生的**原因**：**系统资源不足**、**进程推进顺序不当**。

产生的**必要条件**：**互斥条件**（在一段时间内某资源仅为一个进程所占有）、**不剥夺条件**（进程所获得资源未使用完之前不能被其它进程强行夺走）、**部分分配条件**（又称请求和保持条件：请求新资源时，保持已有资源的拥有权）、**环路等待条件**（存在一种进程资源的循环等待链）。

**处理**方法：**预防死锁**、**避免死锁**、**检测及解除死锁**。

**预防**：通过破坏死锁产生必要条件中的后三条之一来预防死锁的产生。

**避免**：银行家算法（检查资源分配后系统是否处于安全状态，安全状态是指系统处于一种能按某种顺序为每个进程分配其所需的资源，直至最大需求，使每个进程都可以顺利完成的状态）。

**检测**：**简化资源分配图**（分析每个进程所占有的资源数目以及申请的资源数目，找出一个非孤立并且资源申请数量小于系统中已有空闲资源数量的进程，消去它的所有分配边和请求边，使之成为孤立结点。如果能消去所有边，使所有进程都成为孤立结点，则称该图是完全简化的，则不会产生死锁。**死锁定理**：S 为死锁状态的条件是当且仅当 S 状态的资源分配图是不可完全简化的。）、**死锁检测算法**（对于系统中的一组进程，找出其中的各类资源请

求数目均小于系统现在各类空闲资源数目的进程，当其运行结束后，释放所占有的全部资源，使系统中空闲资源数目增加。将其加入可运行结束的进程序列 L 中，直至找不到满足条件的进程为止。如果这组进程中有几个进程不在序列 L 中，则会发生死锁。）

**解除：资源剥夺法、撤销进程法**。

只有当进程提出资源申请且全部进程都进入阻塞状态时，系统才处于死锁状态。

19. 优先权分为静态和动态两种，静态优先权一经确定不再改变，动态优先权会随进程的运行而变化。

20. **高级调度**的主要任务是按给定的原则从外存后备作业中挑选若干作业调入内存，为它们建立进程并分配必要的资源，以使它们获得竞争处理机的权利。 **中级调度**的主要任务是按给定的原则将处于外存对换区中的具备运行条件的就绪进程调入内存，或将内存中处于就绪状态或阻塞状态的进程交换到外存对换区中。 **低级调度**的主要任务是按照某种原则选取一个处于就绪状态的进程，将处理机分配给它。

21. 银行家算法是避免死锁的一种方法，其实现思想是：允许进程动态地申请资源，系统在每次实施资源分配之前，先计算资源分配的安全性，若此次资源分配安全（即资源分配后，系统能按某种顺序来为每个进程分配其所需的资源，直至最大需求，使每个进程都可以顺利完成），便将资源分配给进程，否则不分配资源，让进程等待。

**银行家算法具有较好的理论意义，但在实际系统中却难以实施。其原因是：难以预先获得进程申请的最大资源数；运行过程中进程的个数是不断变化的。所以银行家算法难以用来解决实际中的死锁问题。**

22. 在微机操作系统中，通常把键盘命令分成**内部命令**（程序短小、使用频繁、常驻内存）和外部命令（程序较长，各自独立作为一个文件驻留在磁盘上）两类。

23. 按**命令接口对作业控制方式的不同**可将命令接口分为**联机命令接口**和**脱机命令接口**。

24. 作业的**周转时间：从作业提交到作业完成之间的时间间隔**。**平均周转时间**：多个作业的周转时间的平均值。**带权周转时间**：作业周转时间与作业实际运行时间（对一个作业来说，由于调度，可能周转时间很长，但实际运行时间其实很短）的比。**平均带权周转时间**：多个作业的带权周转时间的平均值。

25. 常用作业调度算法：**先来先服务调度算法、短作业优先调度算法、响应比高者优先调度算法**（响应比=作业响应时间/估计运行时间，其中响应时间为作业进入系统后的等待时间加上估计运行时间，所以：响应比=1+作业等待时间/估计运行时间）、**优先权调度算法**。

26. 在分时操作系统环境下运行的作业通常称为终端型作业。

27. 用户与操作系统之间的接口主要分为**命令接口**和**程序接口**两大类。

28. **系统调用与一般过程调用的主要区别：**

系统调用在本质上是一种过程调用，但它是一种特殊的过程调用，它与一般过程调用的主要区别如下：

1) **运行状态不同**。一般的过程调用，其调用过程和被调用过程都是用户程序，它们都运行在同一系统状态（用户态）下；而系统调用的调用过程是用户程序，运行在用户态。其被调用过程是系统过程，运行在核心态。
2) **进入方式不同**。一般过程调用可以直接通过过程调用语句将控制转移到被调用过程；而执行系统调用时，由于调用过程和被调用过程处于不同系统状态，必须通过访管中断进入。
3) **代码层次不同**。一般过程调用中的被调用程序是用户级程序，而系统调用是操作系统中的代码程序，是系统级程序。

28. **虚拟存储器：基于局部性原理**，在程序装入时可以将程序的一部分放入内存，而将其余

部分放在外存，就可以启动程序执行。在程序执行过程中，当所访问的信息不在内存时，由操作系统将所需要的部分调入内存，然后继续执行程序。另一方面，操作系统将内存中暂时不使用的内容换出到外存上，从而腾出空间存放将要调入内存的信息。从效果上看，这样的**计算机系统好像为用户提供了一个存储容量比实际内存大得多的存储器**，这个存储器称为虚拟存储器。之所以将其称为虚拟存储器，是因为这种存储器实际上并不存在，只是由于系统提供了部分装入、请求调入和置换功能后，给用户的感觉是好像存在一个比实际物理内存大得多的存储器。

29. 为把一个作业装入内存，应按照一定的分配算法从空闲分区表（链）中选出一个满足作业需求的分区分配给作业，如果这个空闲分区的容量比作业要求的容量大，则一部分分配给作业，剩下的一部分仍然留在空闲分区表（链）中，同时需要对空闲分区表（链）中的相关信息进行修改。常用的**分区分配算法**有：

1) **首次适应算法**：又称最先适应算法，该算法要求**空闲分区按地址大小递增的次序排列**。从空闲分区表（链）首开始顺序查找，直到找到第一个满足其大小要求的空闲分区为止。
2) **循环首次适应算法**：又称下次适应算法，该算法不再每次从空闲分区表（链）首开始查找，而是**从上次找到的空闲分区的下一个空闲分区开始查找**。
3) **最佳适应算法：要求空闲分区按容量大小递增的次序排列**，从空闲分区表（链）首开始顺序查找，直到找到第一个满足其大小要求的空闲分区为止。
4) **最坏适应算法：要求空闲分区按容量大小递减的次序排列**，先检查空闲分区表（链）中的第一个空闲分区，若第一个空闲分区小于作业所要求的大小，则分配失败；否则从该空闲分区中划出与作业大小相等的一段内存空间分配给请求者，余下的空闲分区仍然留在空闲分区表（链）中。

30. 当系统发现**所要访问的页不在内存**时，就产生一个**缺页中断**信号。（也就是说，一开始内存中未装入页时，所要访问的页必然不在内存中，也必然要产生缺页中断。）

31. **页面置换（淘汰）算法**（抖动/颠簸：系统把大部分时间用在了页面的调入/调出上）：

1) **最佳置换算法**（Optimal）：从内存中选择永远不再需要的页面或在最长的时间以后才需要的页面将其淘汰。因为页面访问的未来顺序很难精确预测而实现困难，但可以用来评价其他算法的优劣。
2) **先进先出置换算法**：选择在内存中驻留时间最长的页面将其淘汰，即先进入内存的页面，先退出内存。该算法存在一种异常现象，即在某些情况下会出现分配给进程的页面数增多，缺页次数反而增加的怪异现象。（**Belady 现象**）
3) **最近最久未使用算法**（LRU：Least Recently Used）：选择最近一段时间内最长时间没有被访问过的页面将其淘汰。

32. 在执行过程中不能被修改的代码称为可重入代码。

33. 在采用请求分页式存储管理的系统中，地址变换过程可能会因为**地址越界**、**缺页**和**访问权限错误**等原因而产生中断。

34. 设有 8 页的逻辑空间，每页有 1024 字节，它们被映射到 32 块的物理存储区中。那么，逻辑地址的有效位是 **13** 位，物理地址至少是 **15** 位。

分析：每页大小为 1K 字节，8 页的逻辑空间大小为 8K 字节，故逻辑地址的有效位为 13（$2^{13}$=8K）;**在页式系统中，块与页大小相等**，32 块的物理存储区大小为 32K，故物理地址至少为 15（$2^{15}$=32K）。

35. 在存储管理中，**内零头**是指分配给作业的存储空间中未被利用的部分，**外零头**是指系统中无法利用的小存储块。**固定式分区分配中存在内零头，可变式分区分配中存在外零头**（分

配的比需要的多时，一部分分配给作业，剩下的一部分仍然留作系统的空闲分区），**页式虚拟存储系统中存在内零头，段式虚拟存储系统中存在外零头。**

36. 通常，CPU 都是在一条指令执行完后去检查是否有中断请求到达，若有便去响应中断；否则继续执行下一条指令。而缺页中断是在指令执行期间，发现所要访问的指令或数据不在内存时产生和处理的。

37. 有一页式系统，其页表存放在内存中。

1) 如果对内存的一次存取需要 1.5 微秒，问实现一次页面访问的存取时间是多少？
2) 如果系统增加有快表，平均命中率为 85%，当页表项在快表中时，其查找时间忽略为 0，问此时的存取时间为多少？

分析：**若页表存放在内存中，则要实现一次页面访问需要两次访问内存，一次是访问页表，确定所存取页面的物理地址，第二次根据该地址存取页面数据。**

1) 1.5*2=3 微秒；
2) 在系统增加了快表后，在快表中找到页表项的概率是 85%，所以实现一次页面访问的存取时间为：0.85*1.5+（1-0.85）*2*1.5=1.725 微秒。

38. 在一份页存储管理系统中，逻辑地址长度为 16 位，页面大小为 4096 字节，现有一逻辑地址为 2F6AH，且第 0、1、2 页依次存放在物理块 5、10、11 中，问相应的物理地址为多少？

分析：逻辑地址结构：页号（P）、页内位移（W）。由页面大小为 4096 字节，可知页内位移占 12 位，即 0-11 位为页内位移，12-15 位为页号。所以 2F6AH 即为：P(0010)、W(111101101010)，页号为 2，对应物理块为 11，则物理地址为 BF6AH。

**39.** SPOOLing（外部设备同时联机操作）系统在**磁盘**上开辟两个存储区域：输入井、输出井。输入进程模拟脱机输入时的外围控制机，将用户要求的数据从输入机通过输入缓冲区再送到输入井。当 CPU 需要输入数据时，直接将数据从输入井读入内存。输出进程模拟脱机输出时的外围控制机，把用户要求输出的数据先从内存送到输出井，待输出设备空闲时，再将输出井中的数据经过输出缓冲区送到输出设备中。采用以空间换取时间的技术。**SPOOLing 系统是由磁盘中的输入井和输出井，内存中的输入缓冲区和输出缓冲区以及输入进程和输出进程所构成。**

40. 打印机是独享设备，利用 SPOOLing 技术可以讲打印机改造为可供多个用户共享的虚拟设备。

41. 计算机系统为每台设备确定一个编号以便区分和识别设备，这个确定的编号称为设备的绝对号。

42. 中断按照优先级分为如下 5 类：

1) **机器故障中断**：因机器发生错误而产生的中断。如电源故障、内存奇偶校验错等。
2) **访管中断**：由于程序执行了访管指令（系统调用）而产生的中断。如用户程序请求操作系统为其完成某项工作。
3) **程序性中断**：因程序中错误使用指令或数据引起的中断。如定点运算溢出、地址越界、非法指令、缺页中断等。
4) **外部中断**：处理机外部的非通道装置引起的中断。如时钟中断、操作员控制台中断等。
5) **输入/输出中断**：由 I/O 设备引起的中断。如设备传输结束、设备出错等。

43. 借助于**通道**和**中断**技术，可以实现 CPU 与外部设备的并行工作。

44. **缓冲区**由**缓冲首部**和**缓冲体**两部分组成。

45. 设备分配程序分配外部设备时，先分配**设备**，再分配**控制器**，最后分配**通道**。

46. 操作系统在分配设备时，考虑的因素主要有**设备的使用性质**、**设备分配算法**、**设备分配的安全性**。

47. 在计算机系统中，时钟以固定的频率中断 CPU，以增加日历计数或控制系统中的一些定时操作。
48. 设备分配算法：**先请求先服务**、**优先级高者优先**。
49. **文件的逻辑结构**分为两种形式：有结构的**记录式文件**、无结构的**流式文件**。**文件的物理结构**有以下形式：**顺序结构**、**链接结构**、**索引结构**。
50. **磁盘调度算法：先来先服务（FCFS）**、**最短寻道时间优先（SSTF）**、**扫描（SCAN）算法**（扫描算法在磁头当前移动方向上选择与当前磁头所在磁道距离最近的请求作为下一次服务的对象。由于这种算法中磁头移动的规律颇似电梯的运行，故又称为**电梯调度算法**。SCAN 算法具有较好的寻道性能，又避免了饥饿现象；但不利于远离磁头一端的访问请求。）、**循环扫描（CSCAN）算法**（规定磁头单向移动。例如，自里向外移动，当磁头移到最外磁道时立即又返回到最里磁道，如此循环进行扫描。该算法消除了对两端磁道请求的不公平）。
51. 寻道时间是指将磁头从当前位置移动到指定磁道所经历的时间，旋转延迟时间是指定扇区移动到磁头下面所经历的时间。
52. 活动头磁盘的访问时间包括 寻道时间、旋转延迟时间、传输时间。
53. 索引表中每个表项应包含能标识记录的关键字及记录存放地址。
54. 使用文件系统时，通常要显式地进行 OPEN 和 CLOSE 操作，这样做的目的是：显式地 OPEN 操作完成文件的打开功能，它将待访问文件的目录信息读入内存活动文件表中，建立起用户进程与文件的联系。显式 CLOSE 操作完成文件的关闭操作，该命令撤销内存中有关该文件的目录信息，切断用户与该文件的联系；若在文件打开期间，该文件作过某种修改，还应将其写回辅存。取消显式的 OPEN 与 CLOSE 操作使得文件读写的系统开销增加。

【UNIX 操作系统】

1. 在 UNIX System V 中，将 **PCB** 分成**进程表项**和 **U 区**（又称 proc 结构和 user 结构，其中 proc 结构常驻内存），除此之外，管理进程的数据结构还有本进程区表、系统区表。

2. 在 UNIX 的早期版本中，仅为**进程的同步与通信**提供了**软中断信号**（处理异常事件或错误）和**管道机制**（共享文件 pipe 文件，处理进程之间大量信息传送）。在 UNIX System V 中，推出了新的进程间通信机构，简称 IPC（InterProcess Communication）。它由三部分组成：**消息机制**、**共享存储器机制**（是 UNIX 系统中通信速度最高的一种通信机制）和**信号量机制**。

3. 在 UNIX 系统中有三种写方式：把缓冲区数据写入盘块时，进程不需等待操作完成就可继续执行称为**异步写**；需要等待写操作完成才执行称为**同步写**；并不实际写盘、仅设置标志称为**延迟写**。

4. UNIX 操作系统的 shell 是负责解释并执行来自终端的命令的模块。

5. UNIX 系统中，用户通过**系统调用**读取磁盘文件中的数据。

6. UNIX 小部分用汇编、大部分用 C 语言编写，这使得 UNIX 易于移植。

7. 在 UNIX 操作系统中，文件的索引结构存放在 **i 结点**中。

8. 建立无名管道的系统调用是 pipe，创建有名管道的系统调用是 mknod。

9. 为了实现请求调页，UNIX 系统设置了页表、磁盘块描述表、页面数据表和对换使用表。

10. UNIX 文件的目录结构采用树形目录结构，文件的逻辑结构采用流式结构，文件的物理结构采用索引结构。

11. UNIX 的每个目录项由文件名和对应的索引节点号构成，索引节点又可分为内存索引节点和磁盘索引节点。

12. 在 UNIX 系统中，当进程要读/写一个已打开文件时，它依次要访问的数据结构是用户文件描述符表、文件表和内存索引节点表。

13. UNIX 操作系统核心在结构上分为**文件子系统**和**进程控制子系统**。

14. 在 UNIX 系统中，文件共享有基于索引节点的共享方式和基于符号链的共享方式。

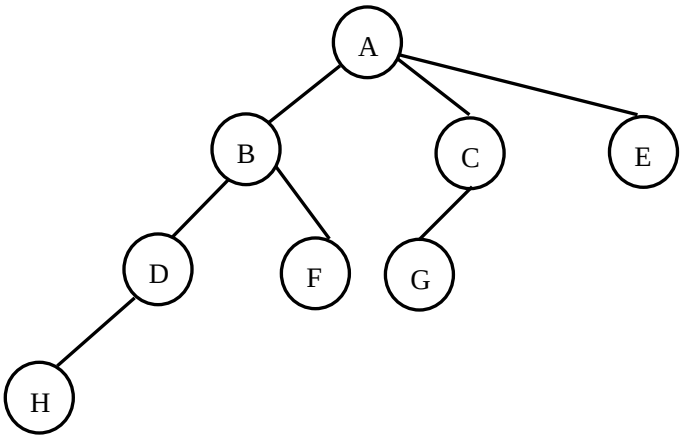
15. 在 UNIX 系统中，采用成组链接法对磁盘空闲块进行管理。

16. 在 UNIX System V 中，一个进程所访问的页面或者在内存中，或者在文件系统中，或者在对换设备上。

17. 在 UNIX 系统中运行下面程序，最多可产生多少个进程？

```
main()
{
    fork();
    fork();
    fork();
}
```

分析：系统调用 fork 的功能是创建一个新进程，新进程运行与其创建者一样的程序，新创建的进程称为子进程，调用 fork 的进程称为父进程，父/子进程都从 fork 调用后的那条语句开始执行。将开始执行时的进程称为 A 进程。执行第一个 fork 调用时，进程 A 创建了进程 B；执行第二个 fork 调用时，进程 A 创建了进程 C，进程 B 创建了进程 D；执行第三个 fork 调用时，进程 A 创建了进程 E，进程 B 创建了进程 F，进程 C 创建了进程 G，进程 D 创建了进程 H。因此，最多可以产生 7 个进程，其进程家族树如下图所示：




18. 在 UNIX 系统中，文件名与文件说明是分开存放的。由文件说明形成的一个数据结构称为索引节点（index node），而相应的文件目录项只由文件名和索引节点号构成。索引节点又称 i 节点，其中存放文件的说明信息。UNIX 系统中的 i 节点不是文件内容的一部分，而是用于文件管理的数据结构。

19. UNIX 进程 0 的主要任务是什么？

分析：当 UNIX 操作系统装入内存后，系统的控制权便由自举程序转到核心程序，即操作系统程序上来。核心首先生成系统进程 0，然后由 0 号进程创建 1 号进程（即 init 进程），进程 1 负责初始化所有新的用户进程。实际上，1 号进程是除了 0 号进程之外所有用户进程的祖先。UNIX 系统的调度与交换是 0 号进程的两部分，它们分别由 swtch 过程和 sched 过程实现。sched 过程把处于外存就绪状态的进程换入内存，swtch 则从就绪队列中寻找一个优先级最高的进程。

所以，**进程 0 的作用是：创建进程 1，进行进程的调度和交换。**

20. 在 UNIX System V 中引起进程调度的情况有以下几种：

1) 当前执行进程因申请资源未得到满足，从而调度过程 sleep 放弃处理机，进入睡眠状态。
2) 为了与其它并发进程保持同步，调用 wait 等过程，从而主动放弃处理机，进入睡眠状态。
3) 当系统从核心态转为用户态时，发现调度标志已设置，即系统中某进程的优先级已高于当前执行进程的优先级，系统调用优先级高的进程执行。
4) 时间片用完，且当前进程的优先级低于其他进程的优先级。
5) 当前进程调用 exit 自我终止。

21. 在 UNIX 系统中，文件的数据存储在离散的磁盘块中，这些**文件的盘块号直接或间接地存放在该文件索引节点 13 个地址项中（直接块 0-直接块 9**(文件逻辑块号小于 10)、**一次间接**(文件逻辑块号大于或等于 10 且小于 266)、**二次间接**(文件逻辑块号大于或等于 266 且小于 65802)、**三次间接**(文件逻辑块号大于或等于 65802)）。

22. 在 UNIX 系统中，一个文件应属于某个用户所有，将这个用户称为该文件的文件主，使用用户标识符（uid）来记录文件主。UNIX 将使用文件的用户分为 3 类：文件主、同组用户（用 gid 表示）和其他用户。每一类用户对文件的操作都有三种不同的存取权限：读、写、执行。文件的存取权限用 9 位二进制描述，例如 111101001 表示：文件主可读、写、执行；同组用户可读、执行；其他用户只能执行。当系统用 ls 命令列目录时，文件权限以标准符号形式表示，上述也可以写为 rwxr-x--x。

23. UNIX 系统为用户提供了两个接口：**面向操作命令的接口 shell** 和**面向编程用户的接口系统调用**。常见的 shell 命令如：login（用户注册）、vi（全屏幕编辑程序）、cp（文件复制）、rm（文件或目录删除）、ls（显示目录）、date（显示或设置系统当前日期和时间）等；常

见的系统调用如：read（文件读）、write（文件写）、open（文件打开）、close（文件关闭）、stat（获取文件属性）、mount（文件系统的安装）、fork（进程创建）等。

24. UNIX 操作系统的进程调度算法是如何使各进程均衡地使用 CPU？

分析：**UNIX 操作系统的进程调度采用可剥夺的动态优先级调度算法。进程的优先级由赋给它的优先数确定，优先数越小，优先级越高。**在该算法中，进程优先数随它的 CPU 时间使用量增加而增加；当进程的 CPU 时间使用量减少时，其优先数也随之减少；同时系统还按一定的时间间隔衰减进程的 CPU 时间使用量。这样，随着进程占用 CPU 时间的逐渐增加，它的优先数也随着增加。相应地，其优先级却在逐渐减小，那么它占用 CPU 的可能性就会减小，从而使该进程占用 CPU 的时间逐渐减少，进程的优先数亦会减小，它相应的优先级就会提高，该进程占用 CPU 的可能性又会增加。于是进程占用 CPU 的时间将逐渐增加……形成一个负反馈，其效果是操作系统会使占用 CPU 时间长的进程减少占用 CPU 的时间，使占用 CPU 时间短的进程增加占用 CPU 的时间，从而使各进程均衡地使用 CPU。

25. 管道通信的基本思想是什么？UNIX 操作系统在管道通信中是如何避免死锁的？

分析：**管道通信以文件系统为基础，在进程之间实现通信。**管道，就是连接两个进程的一个打开的共享文件，该文件专门用于进程之间的通信。发送数据的进程从管道的一端把数据写入管道，接收数据的进程从另一端读出数据，就像一条传送数据的“管道”。管道通信实际上是利用外存来实现进程间的通信，所以具有数据传送量大的特点，但通信速度较慢。在管道的通信过程中，发送进程和接收进程要进行必要的同步与互斥，所以进程可能由于等待而产生死锁。

UNIX 操作系统中采取以下措施来避免死锁：

- 当进程需要读/写等待时，要检查管道的另一端是否已经关闭，如果发现对方已经关闭，则直接返回，不需要等待。
- 当进程需要关闭管道时，要检查管道的另一端是否正处于等待状态；如果是，则要先唤醒对方，然后再关闭管道。

这样进程就不会无休止地等待而发生死锁；

而且，管道通信又可分为无名管道和有名管道。无名管道用于父、子进程之间的通信，而有名管道的适用范围更广。进程无休止地等待不可能发生的事件是产生死锁的必要条件，破坏此条件就可以预防死锁。