作业补充

分时系统和多道程序系统的区别是什么?

在分时系统中，多个用户可以通过自己的终端同时访问使用同一个计算机系统。多道程序系统允许用户同时运行多个程序。所有分时系统都是多道程序系统，但并非所有多道程序系统都是分时系统，因为多道程序系统可能运行在只有一名用户的PC上。

从 Blocked 到 Running 的转换和 从 Ready 到 Running 的转换是否有可能?

从Blocked状态到Running状态的转换是可以的。假设一个进程因 I/O 而阻塞，而 I/O 操作完成了。如果此时 CPU 没有其他任务，该进程可以直接从Blocked状态进入Running状态。从Ready状态到Blocked状态的转换是不可能的。Ready状态的进程不能进行 I/O 操作或任何可能导致Blocked的操作。只有Running状态的进程才会阻塞。

一个多线程的Web服务器，读取文件只能使用阻塞的read系统调用，Web服务器应该使用用户级线程还是内核级线程？为什么?

当工作线程需要从磁盘读取网页时，它会阻塞。如果使用用户级线程，这种操作会导致整个进程阻塞，从而破坏多线程的价值。因此，使用内核级线程是必要的，这样可以让某些线程阻塞而不影响其他线程。

线程可以被抢占吗？如果可以，为什么？如果不可以，为什么?

用户级线程**只有在整个进程的时间片用完时才会被时钟中断抢占**（尽管透明的时钟中断可能会发生）。内核级线程可以单独被抢占。在这种情况下，如果一个线程运行时间过长，时钟会中断当前进程，从而中断当前线程。内核可以选择运行同一个进程中的其他线程。

用户级线程的优缺点是什么?

用户级线程的最大优点是效率高，不需要陷入内核来切换线程。最大的缺点是，如果一个线程阻塞，整个进程都会阻塞。内核级线程可以避免这个问题。

用户级线程是一个进程一个栈还是一个线程一个栈?

每个线程调用自己的过程，因此必须有自己的栈来存储局部变量、返回地址等。这同样适用于用户级线程和内核级线程。

如果线程在内核态实现，能否使用内核信号量对同一个进程中的两个线程进行同步？图个在用户态实现呢？

使用内核级线程时，一个线程可以在信号量上阻塞，内核可以选择运行同一进程中的其他线程。因此，使用信号量没有问题。使用用户级线程时，如果一个线程在信号量上阻塞，内核会认为整个进程被阻塞，从而不再运行它。这会导致进程失败。

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

创建了三个子进程。初始进程分叉后，有两个进程运行，一个父进程和一个子进程。它们各自分叉，创建了另外两个进程。然后所有进程退出。

多级页表的优点是什么？

多级页表通过其层次结构减少了实际需要驻留在内存中的页表页面数量。在程序具有指令和数据局部性的情况下，我们只需要顶级页表（一个页面）、一个指令页面和一个数据页面。

在 UNIX 系统中open系统调用时必须的吗？如果没有使用会有什么后果？

如果没有 `open` 操作，每次读取文件时都需要指定要打开的文件名。系统随后需要获取该文件的 i-node（索引节点），尽管 i-node 可以被缓存。但随之而来的一个问题是何时将 i-node 写回到磁盘。虽然可以通过超时机制来实现，但这会显得有些笨拙，不过也可能会奏效。

硬链接和符号链接的优点？

- 硬链接不需要额外的磁盘空间，只需在 i-node 中增加一个计数器来跟踪指向该文件的硬链接数量。
- 符号链接需要额外的空间来存储指向的目标文件名。符号链接可以指向其他机器上的文件，甚至可以通过互联网访问，而硬链接仅限于指向其所在分区内的文件。

占有和等待条件存在bug：竞争的进程得到了新的资源缺释放了原来的资源，如何解决？

改变请求新资源的语义如下：**如果一个进程请求新资源且资源可用，则它获得该资源并保留已有的资源。如果新资源不可用，则释放所有已有的资源。**在这种情况下，**死锁是不可能的，并且不存在新资源被获取但已有资源丢失的风险。**当然，这只有在可以释放资源的情况下才有效（例如，可以在打印页面之间释放扫描仪，或者在刻录 CD 之间释放 CD 刻录机）。

死锁、活锁以及饥饿的区别

- 死锁发生在一组进程被阻塞等待某个事件，而该事件只能由该组中的其他进程引发。
- 处于活锁中的进程并未被阻塞。相反，它们继续执行，检查某个条件是否成立，而该条件永远不会成立。因此，除了它们持有的资源外，活锁中的进程还会继续消耗宝贵的 CPU 时间。
- 进程的饥饿是由于其他进程的存在以及不断到达的高优先级新进程，使得该进程无法获得资源。与死锁或活锁不同，饥饿可能会自行终止，例如当现有的高优先级进程终止且没有新的高优先级进程到达时。