**死锁**

**什么是死锁？**

所谓死锁，是指**多个进程在运行过程中因争夺资源而造成的一种僵局**，当进程处于这种僵持状态时，若无外力作用，它们都将无法再向前推进。 因此我们举个例子来描述，如果此时有一个线程A，按照先锁a再获得锁b的的顺序获得锁，而在此同时又有另外一个线程B，按照先锁b再锁a的顺序获得锁。如下图所示：

**产生死锁的原因？**

可归结为如下两点：

a. **竞争资源**

系统中的资源可以分为两类：

可剥夺资源，是指某进程在获得这类资源后，该资源可以再被其他进程或系统剥夺，CPU和主存均属于可剥夺性资源；

另一类资源是不可剥夺资源，当系统把这类资源分配给某进程后，再不能强行收回，只能在进程用完后自行释放，如磁带机、打印机等。

产生死锁中的竞争资源之一指的是**竞争不可剥夺资源**（例如：系统中只有一台打印机，可供进程P1使用，假定P1已占用了打印机，若P2继续要求打印机打印将阻塞）

产生死锁中的竞争资源另外一种资源指的是**竞争临时资源**（临时资源包括硬件中断、信号、消息、缓冲区内的消息等），通常消息通信顺序进行不当，则会产生死锁

b. **进程间推进顺序非法**

若P1保持了资源R1,P2保持了资源R2，系统处于不安全状态，因为这两个进程再向前推进，便可能发生死锁

例如，当P1运行到P1：Request（R2）时，将因R2已被P2占用而阻塞；当P2运行到P2：Request（R1）时，也将因R1已被P1占用而阻塞，于是发生进程死锁

**死锁产生的4个必要条件？**

产生死锁的必要条件：

**互斥条件**：进程要求对所分配的资源进行排它性控制，即在一段时间内某资源仅为一进程所占用。

**请求和保持条件**：当进程因请求资源而阻塞时，对已获得的资源保持不放。

**不剥夺条件**：进程已获得的资源在未使用完之前，不能剥夺，只能在使用完时由自己释放。

**环路等待条件**：在发生死锁时，必然存在一个进程--资源的环形链。

**解决死锁的基本方法**

预防死锁：

**资源一次性分配**：一次性分配所有资源，这样就不会再有请求了：（破坏请求条件）

只要有一个资源得不到分配，也不给这个进程分配其他的资源：（破坏请保持条件）

**可剥夺资源**：即当某进程获得了部分资源，但得不到其它资源，则释放已占有的资源（破坏不可剥夺条件）

**资源有序分配法**：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反（破坏环路等待条件）

原文链接：<https://blog.csdn.net/hd12370/article/details/82814348>

**TCP 和 UDP**

**区别**

TCP 是面向连接的，UDP 是面向无连接的

UDP程序结构较简单

TCP 是面向字节流的，UDP 是基于数据报的

TCP 保证数据正确性，UDP 可能丢包

TCP 保证数据顺序，UDP 不保证

**什么是面向连接，什么是面向无连接**

在互通之前，面向连接的协议会先建立连接，如 TCP 有三次握手，而 UDP 不会

**TCP 为什么是可靠连接**

通过 TCP 连接传输的数据**无差错，不丢失，不重复，且按顺序到达**。

TCP **报文头里面的序号能使 TCP 的数据按序到达**

**报文头里面的确认序号能保证不丢包，累计确认及超时重传机制**

TCP 拥有**流量控制及拥塞控制**的机制

TCP 的顺序问题，丢包问题，流量控制都是通过滑动窗口来解决的

拥塞控制时通过拥塞窗口来解决的

**TCP状态中 time\_wait 的作用：**

客户端接收到服务器端的 FIN 报文后进入此状态，此时并不是直接进入 CLOSED 状态，还需要等待一个时间计时器设置的时间。这么做有两个理由：

1 **确保最后一个确认报文段能够到达**。如果 B 没收到 A 发送来的确认报文段，那么就会重新发送连接释放请求报文段，A 等待一段时间就是为了处理这种情况的发生。

2 防止客户端在服务器确认关闭前，再次连接上该服务器，让对应的所有网络包都消失。保证该连接关闭的顺序正确。

原文链接：<https://blog.csdn.net/zhang6223284/article/details/81414149>

hh