**1 重入：**函数被不同的控制流程调用,有可能在第一次调用还没返回时就再次进入该函数，这称为重入。

**2函数可重入**：进程中调用该函数时捕获信号并且在信号处理函数中也调用了该函数，如果在这种场景下不会出问题，那么函数就是可重入的。

**不要在信号处理函数中调用不可重入函数**。

**3** 一个**可重入函数需要满足**：

1. 不使用全局变量或静态变量；
2. 不使用用malloc或者new开辟出的空间；
3. 不调用不可重入函数；
4. 不返回静态或全局数据，所有数据都由函数的调用者提供；
5. 使用本地数据，或者通过制作全局数据的本地拷贝来保护全局数据；

**4** **不可重入函数可能符合以下条件之一**：

1. 调用了**malloc/free函数**，因为malloc函数是用**全局链表**来管理堆的。
2. 调用了**标准I/O库函数**，标准I/O库的很多实现都以不可重入的方式使用全局数据区
3. 可重入体内使用了**静态的数据结构**。

**5 线程安全**：一个函数被称为线程安全的（thread-safe），当且仅当被多个并发线程反复调用时，它会一直产生正确的结果。

6 **可重入函数与线程安全函数的联系与区别**：

函数可以是可重入的，也可以是线程安全的，或者两者皆是，或者两者皆非。

1. **线程安全**是在**多线程**情况下引发的，而**可重入函数**可以在**只有一个线程**的情况下发生。
2. 线程安全不一定是可重入的，而可重入函数则一定是线程安全的。
3. 如果一个函数有全局变量，则这个函数既不是线程安全也不是可重入的。
4. 如果一个函数当中的数据全是自身栈空间的，则这个函数即是线程安全也是可重入的。
5. 如果将**对临界资源的访问加锁**，则这个函数是线程安全的；但如果重入函数的话加锁还未释放，则会产生死锁，因此是不可重入的。

7 **线程安全函数**针对多线程场景而言，在**多线程环境中，必须调用线程安全函数**。**可重入**是针对**异步信号**而言，在**信号处理函数中，必须调用可重入函数**。

1. 线程安全函数能够使不同的线程访问同一块地址空间，而可重入函数要求不同的执行流对数据的操作不影响结果，使结果是相同的。