1.1 进程线程的基本概念（本文讲1.1.1-1.1.3）

1.1.1什么是进程，线程，彼此有什么区别⭐⭐⭐⭐⭐

1.1.2多进程、多线程的优缺点⭐⭐⭐⭐

1.1.3什么时候用进程，什么时候用线程⭐⭐⭐

1.1.4多进程、多线程同步（通讯）的方法⭐⭐⭐⭐

1.1.5进程的空间模型⭐⭐⭐

1.1.6一个进程可以创建多少线程，和什么有关⭐⭐

1.1.7进程线程的状态转换图 什么时候阻塞，什么时候就绪⭐⭐

1.1.8父进程、子进程的关系以及区别⭐⭐⭐⭐⭐

1.1.9什么是进程上下文、中断上下文⭐⭐

1.2 并发，同步，异步，互斥，阻塞，非阻塞的理解

1.2.1并发，同步，异步，互斥，阻塞，非阻塞的理解⭐⭐⭐⭐⭐

1.2.2什么是线程同步和互斥⭐⭐⭐⭐⭐

1.2.3线程同步与阻塞的关系？同步一定阻塞吗？阻塞一定同步吗？⭐⭐⭐

1.3 孤儿进程、僵尸进程、守护进程的概念

1.3.1 基本概念 ⭐⭐

1.3.2 如何创建守护进程：⭐⭐

1.3.3 正确处理孤儿进程、僵尸进程的方法⭐⭐⭐⭐⭐

1.1 进程线程的基本概念

1.1.1什么是进程，线程？也就是问彼此有什么区别？

答：进程是资源（CPU、内存等）分配的基本单位，线程是CPU调度和分配的基本单位（程序执行的最小单位）。

1）当我们运行一个程序的时候，系统就会创建一个进程，并分配地址空间和其他资源，最后把进程加入就绪队列直到分配到CPU时间就可以正式运行了。

2）线程是进程的一个执行流，有一个初学者可能误解的概念，进程就像一个容器一样，包括程序运行的程序段、数据段等信息，但是进程其实是不能用来运行代码的，真正运行代码的是进程里的线程。

3）那么，来看看我们最熟悉的main()函数，我们既可以认为这是一个进程，也可以认为是一个线程。我们都知道，在C/C++中main函数是程序入口，所以准确来说main函数是程序的主线程。然而很神奇的地方在于，当系统在执行main函数的时候，main函数又是一个独立的进程，我们可以在main函数里创建子进程，也可以创建子线程。

4）在main函数里创建的多个子线程中，每个线程有自己的堆栈和局部变量，但多个线程也可共享同个进程下的所有共享资源，因此我们经常可以创建多个线程实现并发操作，实现更加复杂的功能。

1.1.2 多进程、多线程的优缺点

解析：为了理解多进程、多线程各自的优缺点之前，我们需要先了解进程和线程最大的区别和联系，一个进程由PCB（进程控制块）、数据段、代码段组成，进程本身不可以运行程序，而是像一个容器一样，先创建出一个主线程，分配给主线程一定的系统资源，这时候就可以在主线程开始实现各种功能。当我们需要实现更复杂的功能时，可以在主线程里创建多个子线程，跟人多好干活的道理一样，多个线程在同一个进程里，利用这个进程所拥有的系统资源合作完成某些功能。

理解了这些知识点，再来理解各自优缺点就很容易了。

1）多进程更健壮，一个进程死了不影响其他进程，子进程死了也不会影响到主进程，毕竟系统会给每个进程分配独立的系统资源。多线程比较脆弱，一个线程崩溃很可能影响到整个程序，因为多个线程是在一个进程里一起合作干活的。

2） 进程性能大于线程，每个进程独立地址空间和资源，而多个线程是一起共享了同个进程里的空间和资源，结果就很明显了，线程的性能上限一定比不上进程。

3） 正因为进程性能大于线程。所以这也引发了另一重要知识点，创建多进程的系统花销远大于创建多线程。

4）多进程通讯因为需要跨越进程边界，不适合大量数据的传送，更适合小数据或者密集数据的传送。而多线程无需跨越进程边界，适合各线程间大量数据的传送，甚至还有很重要的一点，多线程可以共享同一进程里的共享内存和变量哦。

5） 多进程逻辑控制比多线程复杂，需要与主进程做好交互。根据上面几点，我们不难知道多进程是“要用来做大事”的，而多线程是“各自做件小事，合作完成大事”。所以要做大事自然就需要更复杂的逻辑控制，不像做小事那么目标明显。

6）虽然多线程逻辑控制比较简单，但是却需要复杂的线程同步和加锁控制等机制。

7）最后的一点，可能比较少见，我们可以通过增加CPU的数量来增加进程的数量，但增加不了线程的数量，即增加CPU无法提高线程数量，线程数量由进程的空间资源和线程本身栈大小确定，详情见1.1.6小节。

1.1.3 什么时候用进程，什么时候用线程

解析：还是同一个思想，进程是“要用来做大事”的，而线程是“各自做件小事，合作完成大事”，结合上节新鲜出炉的优缺点我们就很好理解什么时候用进程或者线程了。

1）创建和销毁较频繁使用线程，因为创建进程花销大嘛。

2）需要大量数据传送使用线程，因为多线程切换速度快，不需要跨越进程边界。

3）并行操作使用线程。线程是为了实现并行操作的一个手段，也就是刚才说的需要多个并行操作“合作完成大事”，当然是使用线程啦。

4）最后可以总结为：安全稳定选进程；快速频繁选线程；

1.1.4 多进程、多线程同步（通讯）的方法

答：当我们在使用系统编程时，就会遇到多进程、多线程编程，所以必须知道在多个进程、多个线程之间都有什么常见的通讯机制，这也是嵌入式面试中高频问题之一。

进程间通讯：

（1）管道/无名管道（2）信号（3）共享内存（4）消息队列（5）信号量（6）socket

注意：临界区则是一种概念，指的是访问公共资源的程序片段，并不是一种通信方式。

线程通讯：

（1）信号量（2）读写锁（3）条件变量（4）互斥锁（5）自旋锁

提问：互斥锁与信号量的区别？

答：互斥锁用于线程的互斥，信号量用于线程的同步。这是互斥锁和信号量的根本区别，也就是互斥和同步之间的区别。同时互斥锁的作用域仅仅在于线程，信号量可以作用于线程和进程。

1.1.5 进程的空间模型

解析：32位系统中，当系统运行一个程序，就会创建一个进程，系统为其分配4G的虚拟地址空间，其中0-3G是用户空间，3-4G是内核空间，具体如图1-2，内核空间是受保护的，用户不能对该空间进行读写操作，否则可能出现段错误。其中栈空间有向下的箭头，代表数据地址增加的空间是往下的，新的数据的地址的值反而更小，堆空间则是往上。

1.1.6 一个进程可以创建多少线程，和什么有关

答：一个进程创建线程的个数由虚拟内存和分配给线程的调用栈大小决定。

由1.1.5小节我们已经知道创建一个进程，系统会分配4G的虚拟内存，其中1G是内核空间，只有3G是用户空间，也就是我们可以利用来创建线程的空间大小，一个线程的栈的大小可以通过ulimit -s指令来查看，一般大多是8M-10M。

举个例子，这里不放假设我们创建一个线程的栈需要占用10M内存，因此在3G的空间大概可以创建300个线程。此时如果将线程栈大小增加到20M，那么个数就将减少。