多进程和多线程及共享内存学习笔记

１．进程和线程

进程和线程是多进程和多线程实现的基础，要知道什么是多线程和进程就要了解什么是进程和线程

1、进程 进程是程序在计算机上的一次执行活动。是一个具有一定独立功能的程序在某个数据集合的一次运行活动。程序是静态的，进程是动态的。 进程具有的特征：

动态性：进程是程序的一次执行过程，是临时的，有生命期的，是动态产生，动态消亡的；

并发性：任何进程都可以同其他进程一起并发执行（**所以要注意并行操作准确性的处理**）；

独立性：进程是系统进行资源分配和调度的一个独立单位；

结构性：进程由程序、数据和进程控制块三部分组成。程序用于描述进程要完成的功能，是控制进程执行的指令集；数据集合是程序在执行时所需要的数据和工作区；程序控制块，包含进程的描述信息和控制信息，是进程存在的唯一标志。

2、线程 线程是进程内的并发，是进程中的一个执行控制单元。

使用线程的好处

1、使用线程可以把占据长时间的程序中的任务放到后台去处理

2、用户界面可以更加吸引人，这样比如用户点击了一个按钮去触发某些事件的处理，可以弹出一个进度条来显示处理的进度 （**这就是传说中的进度条！？？？**）

3、程序的运行速度可能加快

4、在一些等待的任务实现上如用户输入、文件读写和网络收发数据等，使用线程我们可以释放一些珍贵的资源如内存占用等等

３、小结

进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

２．多进程和多线程

１、多进程 同一个时间里，同一个计算机系统中允许两个或两个以上的进程处于并行状态，这是多进程。多进程带来的好处是你可以边发消息，与此同时甚至可以将下载的文档打印出来，而这些任务之间丝毫不会相互干扰。但并行需要解决的问题是通常并行的进程比CPU数量多得多，而原则上一个CPU只能分配给一个进程。以便运行这个进程。实现并发技术最常见的就是时间片轮转调度算法，即在操作系统的管理下，所有正在运行的进程轮流使用CPU，每个进程允许占用CPU的时间非常短(比如1毫秒)，这样用户根本感觉不出来 CPU是在轮流为多个进程服务，就好象所有的进程都在不间断地运行一样，给用户的感觉就是并行。但实际上在任何一个时间内有且仅有一个进程占有CPU。**而且随着进程数量的增多效率会越来越低**。

２、多线程 一个进程中可以有多条执行路径同时执行，一个线程就是进程中的一条执行路径。

在早期的操作系统中并没有线程的概念，进程是能拥有资源和独立运行的最小单位，也是程序执行的最小单位。它相当于一个进程里只有一个线程，进程本身就是线程。所以线程有时被称为轻量级进程。后来，随着计算机的发展，对多个任务之间上下文切换的效率要求越来越高，就抽象出一个更小的概念——线程，一般一个进程会有多个(也可是一个)线程。

3.多进程和多线程的比较

对比类别	多进程	多线程	总结
数据共享、同步	数据共享复杂，需要用IPC；数据是分开的，同步简单	因为共享进程数据，数据共享简单，但也是因为这个原因导致同步复杂	各有优势
内存、CPU	占用内存多，切换复杂，CPU利用率低	占用内存少，切换简单，CPU利用率高	线程占优
创建销毁、切换	创建销毁、切换复杂，速度慢	创建销毁、切换简单，速度很快	线程占优
编程、调试	编程简单，调试简单	编程复杂，调试复杂	进程占优
可靠性	进程间不会互相影响	一个线程挂掉将导致整个进程挂掉	进程占优
分布式	适应于多核、多机分布式；如果一台机器不够，扩展到多台机器比较简单	适应于多核分布式	进程占优

综上所述：

1）需要频繁创建销毁的优先用线程

这种原则最常见的应用就是Web服务器了，来一个连接建立一个线程，断了就销毁线程，要是用进程，创建和销毁的代价是很难承受的,**例如笔者之前做ｓｏｃｋｅｔ编程时，使用ｆｏｒｋ开子进程进行并发作业时，每当创建和销毁是ｃｐｕ占用率飞速提高**

2）需要进行大量计算的优先使用线程

所谓大量计算，当然就是要耗费很多CPU，切换频繁了，这种情况下线程是最合适的（**可能是为了省电？**）。

3）强相关的处理用线程，弱相关的处理用进程

一般的Server需要完成如下任务：消息收发、消息处理。“消息收发”和“消息处理”就是弱相关的任务（**比如说ｓｏｃｋｅｔ简易聊天室**），而“消息处理”里面可能又分为“消息解码”、“业务处理”，这两个任务相对来说相关性就要强多了。因此“消息收发”和“消息处理”可以分进程设计，“消息解码”、“业务处理”可以分线程设计。

4）可能要扩展到多机分布的用进程（**比如之前做的最长字符串？？？？**），多核分布的用线程

5）都满足需求的情况下，用顺手的

４．共享内存

1.什么是共享内存?

共享内存就是允许两个或多个进程共享一定的存储区。就如同 malloc() 函数向不同进程返回了指向同一个物理内存区域的指针。当一个进程改变了这块地址中的内容的时候，其它进程都会察觉到这个更改。因为数据不需要在客户机和服务器端之间复制，数据直接写到内存，不用若干次数据拷贝，所以这是最快的一种IPC。

（**共享内存没有任何的同步与互斥机制，所以要使用信号量来实现对共享内存的存取的同步**）

２．与共享内存有关的函数

所有的函数共用头文件

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

创建共享内存——>shmget() 函数

```
int shmget(key_t key, size_t size, int shmflg);
                //成功返回共享内存的ID,出错返回-1
```

操作共享内存———>shmctl()函数

```
int shmctl(int shm_id, int cmd, struct shmid_ds *buf);
                //成功返回0，出错返回-1
```

挂接操作———>shmat()函数

创建共享存储段之后，将进程连接到它的地址空间

```
void *shmat(int shm_id, const void *shm_addr, int shmflg);
                    //成功返回指向共享存储段的指针，出错返回-1
```

分离操作———>shmdt()函数

该操作不从系统中删除标识符和其数据结构，要显示调用shmctl(带命令IPC_RMID)才能删除它

```
int shmdt(const void *shmaddr);
            //成功返回0，出错返回-1
```