那些不能遗忘的知识点回顾——操作系统系列（笔试面试高频题） - 闻波 - 博客园

星期四, 三月 21, 2019 9:37 下午

已剪辑自: https://www.cnblogs.com/webary/p/4782903.html

有那么一些零碎的小知识点，偶尔很迷惑，偶尔被忽略，偶然却发现它们很重要，也是各大笔试和面试高频出现考点。这段时间正好在温习这些，就整理在这里，一起学习一起提高！后面还会继续补充。

——前言

1.进程和线程

线程是指进程内的一个执行单元,也是进程内的可调度实体.

与进程的区别:

(1)地址空间:进程内的一个执行单元;进程至少有一个线程;它们共享进程的地址空间(也有少量自己的地址空间);而进程有自己独立的地址空间(多个进程之间一般不会共享地址空间);

(2)资源拥有:进程是资源拥有的单位,同一个进程内的线程共享进程的资源

(3)线程是处理器调度和分派的基本单位.

(4)二者均可并发执行.多线程程序的并发性高。

(5)进程的切换代价远高于线程，同步和通信的实现也比线程复杂。

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，但线程之间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，需要用多线程。

2.Unix和windows进程间通信的主要方式

linux系统IPC:

管道(pipe)：管道是一种半双工的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用。进程的亲缘关系通常是指父子进程关系。

命名管道 (named pipe)： 命名管道也是半双工的通信方式，但是它允许无亲缘关系进程间的通信。

信号量(semophore)： 信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。

消息队列(message queue)： 消息队列是由消息的链表，存放在内核中并由消息队列标识符标识。消息队列克服了信号传递信息少、管道只能承载无格式字节流以及缓冲区大小受限等缺点。

信号 (sinal)：信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。

共享内存(shared memory)： 共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。共享内存是最快的 IPC 方式，它是针对其他进程间通信方式运行效率低而专门设计的。它往往与其他通信机制，如信号量，配合使用，来实现进程间的同步和通信。

套接字(socket)： 套解口也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同及其间的进程通信。

windows系统IPC:

剪贴板(Clipboard)：当用户在应用程序中执行剪切或复制操作时，应用程序将选定的数据以一个或多个标准或应用程序定义的格式放在剪贴板中。

WM_COPYDATA消息：当一个应用向另一个应用传送数据时，发送方只需使用调用SendMessage函数，接收方只需像处理其它消息那样处理WM_COPYDATA消息，这样收发双方就实现了数据共享，它在底层实际上是通过文件映射来实现的。

文件映射(File Mapping)：使进程把文件内容当作进程地址区间一块内存那样来对待。只需简单的指针操作就可读取和修改文件的内容。 允许多个进程访问同一文件映射对象，各个进程在它自己的地址空间里接收内存的指针，通过使用这些指针，不同进程就可以读写文件的内容，实现了对文件中数据的共享。

共享内存(Shared Memory)是文件映射的一种特殊情况进程在创建文件映射对象时用0xFFFFFFFF来代替文件句柄(HANDLE)，就表示了对应的文件映射对象是从操作系统页面文件访问内存，其它进程打开该文件映射对象就可以访问该内存块。由于共享内存是用 文件映射实现的， 所以它也有较好的安全性，也只能运行于同一计算机上的进程之间。

动态数据交换(DDE)：是使用共享内存在应用程序之间进行数据交换的一种进程间通信形式。应用程序可以使用DDE进行一次性数据传输，也可以当出现新数据时，通过发送更新值在应用程序间动态交换数据。DDE和剪贴板一样既支持标准数据格式(如文本、位图等)，又可以支持自己定义的数据格式。但它们的数据传输机制却不同，一个明显区别是剪贴板操作几乎总是用作对用户指定操作的一次性应答，如从菜单中选择Paste命令。尽管DDE也可以由用户启动，但它继续发挥作用一般不必用户进一步干预。可以发生在单机或网络中不同计算机的应用程序之间。

邮件槽(Mailslot)：提供进程间单向通信能力，任何进程都能建立邮件槽成为邮件槽服务器。其它进程称为邮件槽客户，可以通过邮件槽的名字给邮件槽服务器进程发送消息。进来的消息一直放在邮件槽中，直到服务器进程读取它为止。一个进程既可以是邮件槽服务器也可以是邮件槽客户，因此可建立多个 邮件槽实现进程间的双向通信。

管道(pipe)：同上linux系统＆命名管道

套接字（Sockets ）：同上linux系统

3.死锁

死锁是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的僵局，若无外力作用，它们都将无法推进下去。

产生死锁的四个**必要条件**:

1.互斥条件：一段时间内某资源只由一个进程占有。

2.请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

3.不剥夺条件:进程已获得资源，在未使用完之前，不能强行剥夺。

4.循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

预防死锁：需要打破必要条件的2，3，4中之一，由于施加的限制条件较严格，可能导致系统资源利用率和系统吞吐量降低。

避免死锁：施加的限制条件较弱，使系统一直处于安全状态。比如银行家算法。

检测死锁：资源分配图、死锁定理。

解除死锁：剥夺起源、撤销进程。

4.windows下什么线程优先级最高

SetThreadPriority 设置指定线程的优先级：

BOOL SetThreadPriority(HANDLE hThread, int nPriority);

参数说明：

hThread 要设置的线程句柄

nPriority 优先级别参数 可设置为一下参数

THREAD_PRIORITY_ABOVE_NORMAL 比一般优先级高一个等级

THREAD_PRIORITY_BELOW_NORMAL 比一般低一个等级

THREAD_PRIORITY_HIGHEST 比一般高2个等级（最高）

THREAD_PRIORITY_IDLE 空闲

THREAD_PRIORITY_LOWEST 比一般低2个等级（最低）

THREAD_PRIORITY_NORMAL 一般等级

THREAD_PRIORITY_TIME_CRITICAL 实时

5.linux下fork函数

在fork()的调用处，创建一个子进程，并将整个父进程空间会原模原样地复制到子进程中，包括指令，变量值，程序调用栈，环境变量，缓冲区等。fork调用仅仅被调用一次，却能够返回两次，它可能有三种不同的返回值：

（1）在父进程中，fork返回新创建子进程的进程ID；

（2）在子进程中，fork返回0；

（3）如果出现错误，fork返回一个负值；

在fork函数执行完毕后，如果创建新进程成功，则出现两个进程，一个是子进程，一个是父进程。在子进程中，fork函数返回0，在父进程中，fork返回新创建子进程的进程ID。我们可以通过fork返回的值来判断当前进程是子进程还是父进程。

fork出错可能有两种原因：

1）当前的进程数已经达到了系统规定的上限，这时errno的值被设置为EAGAIN。

2）系统内存不足，这时errno的值被设置为ENOMEM。

创建新进程成功后，系统中出现两个基本完全相同的进程，这两个进程执行没有固定的先后顺序，哪个进程先执行要看系统的进程调度策略。

```
#include <unistd.h>
#include <stdio.h>
int main()
{
   int i=0;
   for(i=0;i<3;i++) {
       pid_t fpid = fork();
       if(fpid==0)
          printf("son\n");
        else
           printf("father\n");
   }
    return 0;
}
```

对于这种N次循环的情况，执行printf函数的次数为**2*（2^N-1）**次，创建的子进程数为**2^N-1**个。输出中没有换行时缓冲区也会被复制，参见：http://www.oschina.net/question/195301_62902。

6.程序什么时候使用多线程好，什么时候单线程效率高

1．耗时的操作使用线程，提高应用程序响应速度

2．并行操作时使用线程，如C/S架构的服务器端并发线程响应用户的请求

3．多CPU系统中，使用线程提高CPU利用率

4．改善程序结构。一个既长又复杂的进程可以考虑分为多个线程，成为几个独立或半独立的运行部分，这样的程序会利于理解和修改。其他情况都使用单线程。

7.线程间通信

互锁函数、临界段、内核对象（事件对象、互斥对象、信号量）

8.进程状态转换

在操作系统中，进程一般有三种基本状态：运行状态，就绪状态和等待状态。

1）就绪——执行：对就绪状态的进程，当进程调度程序按一种选定的策略从中选中一个就绪进程，为之分配了处理机后，该进程便由就绪状态变为执行状态；

2）执行——等待：正在执行的进程因发生某等待事件而无法执行，如进程提出输入/输出请求而变成等待外部设备传输信息的状态，进程申请资源（主存空间或外部设备）得不到满足时变成等待资源状态，进程运行中出现了故障（程序出错或主存储器读写错等）变成等待干预状态等等；

3）等待——就绪：处于等待状态的进程，在其等待的事件已经发生，如输入/输出完成，资源得到满足或错误处理完毕时，处于等待状态的进程并不马上转入执行状态，而是先转入就绪状态，然后再由系统进程调度程序在适当的时候将该进程转为执行状态；

4）执行——就绪：正在执行的进程，因时间片用完而被暂停执行，或在采用抢先式优先级调度算法的系统中,当有更高优先级的进程要运行而被迫让出处理机时，该进程便由执行状态转变为就绪状态。

9.内存地址：虚拟地址-线性地址-物理地址的区别与联系

x86平台下的系统采用分段机制与分页机制对地址进行转换，其中分段机制把一个虚拟地址转换成线性地址；分页机制把一个线性地址转换成物理地址。

参考资料：

《计算机操作系统（第三版）》，西安电子科技大学出版社，汤小丹等

linux中fork（）函数详解（原创！！实例讲解）

一个fork的面试题

————————————————————————————我是分割线
————————————————————————

操作系统系列的暂时整理到这里吧，如果读者发现还有哪些这方面的经典常考知识点也请指出，待续~