# 面试学习笔记

## 线程与进程的区别和联系?

线程是否具有相同的堆栈? dll是否有独立的堆栈?

答案：进程是死的，只是一些资源的集合，真正的程序执行都是线程来完成的，程序启动的时候操作系统就帮你创建了一个主线程。每个线程有自己的堆栈。 DLL中有没有独立的堆栈，这个问题不好回答，或者说这个问题本身是否有问题。因为DLL中的代码是被某些线程所执行，只有线程拥有堆栈，如果DLL中的代码是EXE中的线程所调用，那么这个时候是不是说这个DLL没有自己独立的堆栈？如果DLL中的代码是由DLL自己创建的线程所执行，那么是不是说DLL有独立的堆栈？以上讲的是堆栈，如果对于堆来说，每个DLL有自己的堆，所以如果是从DLL中动态分配的内存，最好是从DLL中删除，如果你从DLL中分配内存，然后在EXE中，或者另外一个DLL中删除，很有可能导致程序崩溃

一.进程和线程基本概念

1.进程：进程是操作系统中资源分配的基本单位。进程是操作系统对正在运行程序的一种抽象，可以将进程看作程序的一次运行。

2.线程：线程是操作系统中调度执行的基本单位。一个线程是一个“执行流”，每个线程之间都可以按照顺序执行自己的代码，多个线程“同时”执行多份代码。

二.进程和线程之间的区别与联系

1.一个进程可以包含多个线程，线程在进程的内部。

2.进程之间的资源是独立的，线程之间的资源则是共享的。

每个进程都有独立的虚拟地址空间，也有之间独立的文件描述符表，同一进程的多个线程之间则共用这一份虚拟地址空间和文件描述符表。

3.进程是操作系统中资源分配的基本单位，线程是操作系统中调度执行的基本单位。

4.多个进程同时执行时，如果一个进程崩溃，一般不会影响其他进程，而同一进程内的多个线程之间，如果一个线程崩溃，很可能使得整个进程崩溃。

5.进程的上下文切换速度比较慢，而线程的上下文切换速度比较快。

6.进程的创建/销毁/调度开销大，线程的创建/销毁/调度开销相对少很多。

## (a++)\*(a++)

int main()

{

int a = 5;

int b = 0;

b = (a++)\*(a++);

printf("b:%d\n", b);

return 0;

}

结果是２５

为什么不是30呢，因（a++）\*(a++)是一个乘法表达式，(a++)处理需要两条指令，第一条是获取a的值，++在第二条指令,而这个表达式在（a++）第一条指令就已经获取到值，所以++操作才会被置后处理．

实际上这段c代码被编译成汇编就成这样了：

　　movl $5, 28(%esp)

movl 28(%esp), %eax

imull 28(%esp), %eax

movl %eax, 24(%esp)

addl $1, 28(%esp)

addl $1, 28(%esp)

再成c翻译一下：

a=5:

b=a\*a;

a=a+1;

a=a+1;

或许其他编译器会有不同的解释．

SPI，IIC

IO模拟SPI与转有通讯端口区别

堆和栈的区别

函数调用压栈

介绍一个驱动项目

多态

编译时多态，运行时多态

派生函数重写加不加override的区别

重载运算符

指针和引用的区别

声明的定义的区别

析构函数是否是虚函数

函数后面加const表示什么

vector和链表区别

unordered和mapde 底层原理

快排以及非递归写法