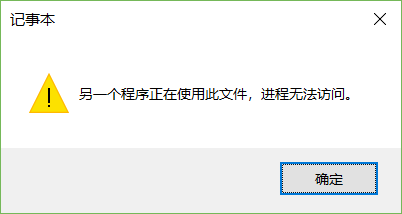
1. 用Word 和 notepad 打开并编辑相同文件, 看对文件的修改是否有冲突, 请说明在你电脑上表现出的情况并尝试解释说明。

用word与记事本打开同一文件（打开顺序无关），在记事本上进行编辑并保存，出现如下对话框：



点击“确定”后弹出“另存为”对话框。

反过来，在word上编辑并保存，无任何异常。

解释：word和记事本均拥有对文件的读权限，读操作之间不冲突。Word打开文件时禁止了其他应用对这个文件的写权限（包括另一次用word打开相同文件），记事本在保存文件时才去检查文件的写权限，因此发生冲突。

2. (1)在主线程中循环创建子线程，看在你的系统里最多能创建多少个，并尝试分析。

809或810，再创建线程时返回代码11。

查看PTHREAD\_THREADS\_MAX=2019。

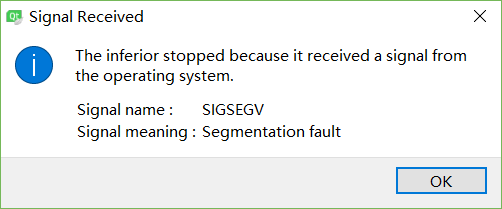
(2)在主线程中创建2个子线程，编写任务使得子线程发生栈溢出，并对发生的现象进行分析说明。

如果溢出不超过局部变量分配地址范围，只会发生简单的数据覆盖，无论数据位于子线程内或子线程外均如此。例如

int p, x[10], y[10]

y[15]对应于x[5]，y[20]对应于p。说明局部变量在栈中是由高地址至低地址连续分配的。

如果栈溢出至不可读或写的位置，debug模式下程序会失败退出，且程序输出文本长度不确定：



而run模式下，在pthread\_join操作前会正常输出，子线程出现异常时无报错，而对出现异常的子线程执行pthread\_join操作会使主线程挂起，从而无法执行后续输出操作。其他线程的输出不会受到影响。

(3) 略。

(4) 编写如下几个多线程程序，让2个线程共享全局计数器，每个线程循环执行任务，每做完一个任务对计数器进行增1，并打印形如 Thread X finished No.Y job 的信息，X代表线程编号，Y代表当前的全局计数值

<1> 访问全局计数器时不使用任何同步。

<2> 访问全局计数器时使用互斥锁。

见代码及output.txt。由于2个子线程难以观察异常，本实验中创建了50个子线程。

有锁时就实验结果而言是按序输出的，尽管各线程的计算进度可能不同。

无锁时几乎总是有不按序输出的情况，在输出示例中有一些相隔45-47号的行互换，或某行实际输出于后面相隔较远的位置。互换的行对应的线程一般不相同。