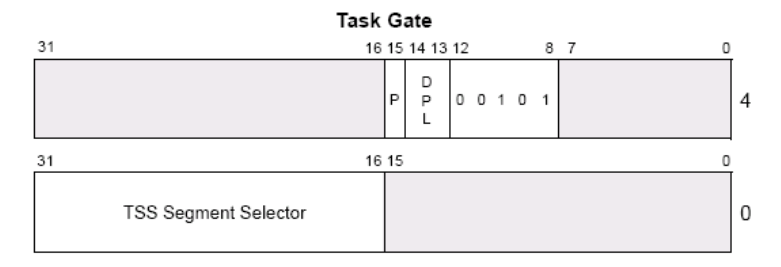
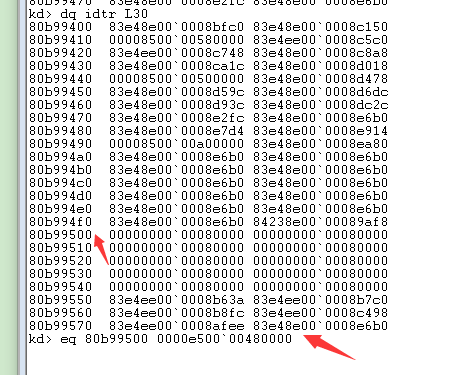
# 任务门

为什么要学TSS和任务门？因为8号错误是双重错误，为了定位真正触发错误的eip，需要用到这些知识。

上次课学了TSS任务状态段，这节课的任务门非常简单，只是在TSS进R0的基础上，在IDT表构造一个任务门，然后通过中断调用任务门进R0.代码也是在上节课的基础上修改。



首先构造一个，还是在熟悉的 int 0x20。任务门描述符放在IDT表，TSS选择子填48，待会去GDT 48构造一个TSS描述符。所以任务门描述符就可以确定是 0000e500`00480000

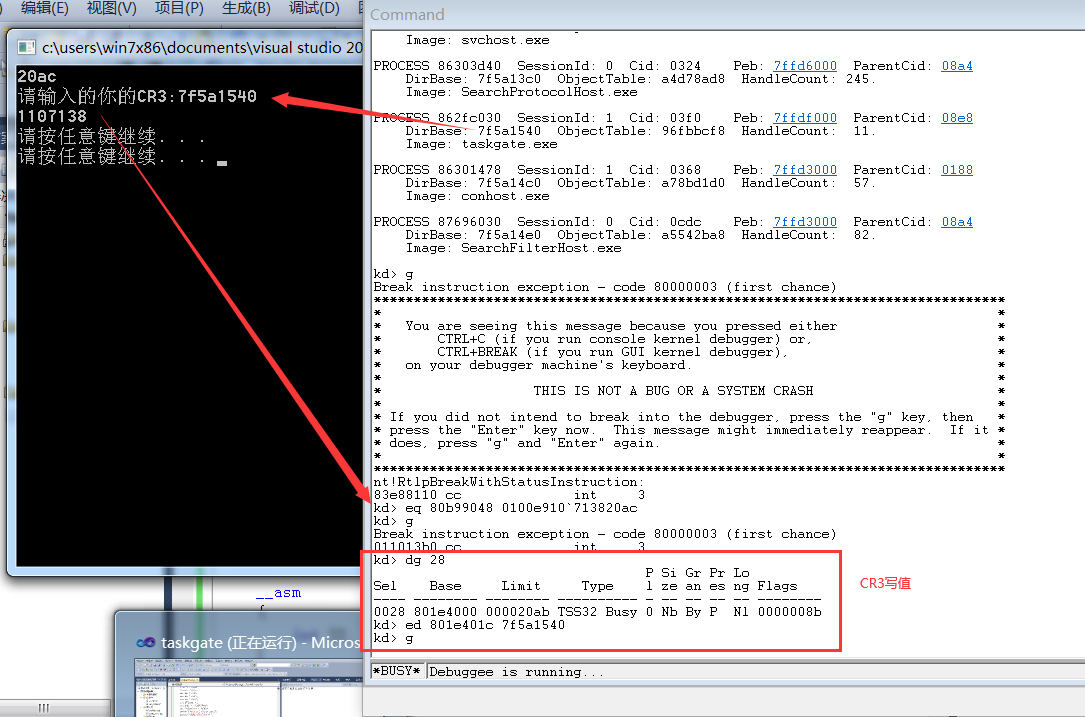


eq 80b99500 0000e500`00480000

代码和TSS课上的代码基本一样，唯一区别是call far改成int 32

|  |
| --- |
| #include "stdafx.h"  #include <Windows.h>  struct \_KiIoAccessMap  {  UCHAR DirectionMap[32]; //0x0  UCHAR IoMap[8196]; //0x20  };  typedef struct \_KTSS  {  USHORT Backlink; //0x0  USHORT Reserved0; //0x2  ULONG Esp0; //0x4  USHORT Ss0; //0x8  USHORT Reserved1; //0xa  ULONG NotUsed1[4]; //0xc  ULONG CR3; //0x1c  ULONG Eip; //0x20  ULONG EFlags; //0x24  ULONG Eax; //0x28  ULONG Ecx; //0x2c  ULONG Edx; //0x30  ULONG Ebx; //0x34  ULONG Esp; //0x38  ULONG Ebp; //0x3c  ULONG Esi; //0x40  ULONG Edi; //0x44  USHORT Es; //0x48  USHORT Reserved2; //0x4a  USHORT Cs; //0x4c  USHORT Reserved3; //0x4e  USHORT Ss; //0x50  USHORT Reserved4; //0x52  USHORT Ds; //0x54  USHORT Reserved5; //0x56  USHORT Fs; //0x58  USHORT Reserved6; //0x5a  USHORT Gs; //0x5c  USHORT Reserved7; //0x5e  USHORT LDT; //0x60  USHORT Reserved8; //0x62  USHORT Flags; //0x64  USHORT IoMapBase; //0x66  struct \_KiIoAccessMap IoMaps[1]; //0x68  UCHAR IntDirectionMap[32]; //0x208c  }KTSS;  KTSS tss={0};  char bufEsp0[0x2000]={0};  char bufEsp3[0x2000]={0};  void \_\_declspec(naked) test()  {  \_\_asm  {  int 3;  pushfd;  pop eax;  or eax,0x4000; // eflags.NT = 1  push eax;  popfd;  iretd;  }  }  int \_tmain(int argc, \_TCHAR\* argv[])  {  memset(bufEsp0,0,0x2000);  memset(bufEsp3,0,0x2000);  tss.Esp0 = (ULONG)bufEsp0 + 0x1FF0;  tss.Esp = (ULONG)bufEsp3 + 0x1FF0;  tss.Ss0 = 0x10;  tss.Ss = 0x10;  tss.Cs = 0x8;  tss.Ds = 0x23;  tss.Es = 0x23;  tss.Fs = 0x30;  tss.EFlags = 2;  tss.Eip = (ULONG)test;  tss.IoMapBase = 0x20ac;  printf("%x\r\n",sizeof(tss));  printf("请?输?入?的?你?的?CR3:");  DWORD dwCr3 = 0;  scanf("%x",&dwCr3);  tss.CR3 = dwCr3;  printf("%x\r\n",&tss);  system("pause");    char bufcode[]={0,0,0,0,0x48,0};    \_\_asm  {  int 32;  }  system("pause");  return 0;  } |

操作步骤也一样，拿一下CR3，然后根据TSS数组的地址，构造GDT48的TSS描述符：

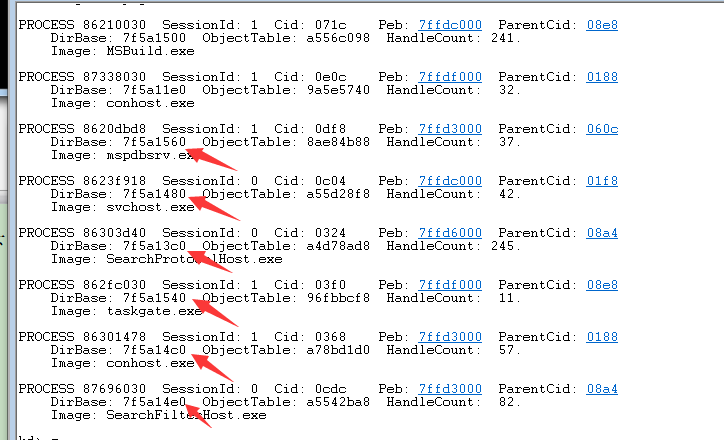


修复一下CR3，g返回就行了。

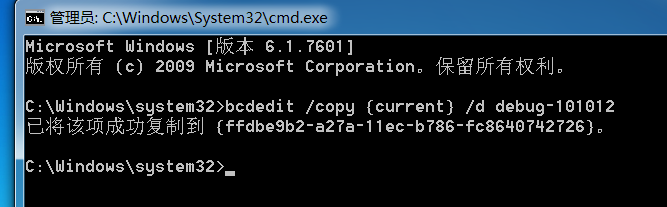
# 101012环境配置

Win7默认是29912分页，下面把它改成101012

29912分页的时候，!process 0 0 看CR3，可以发现是以0x20 单位增减的：



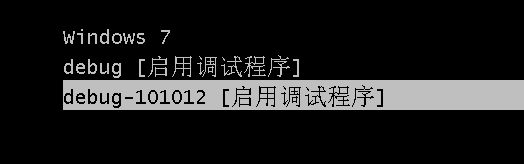
接下来，用bcdedit 创建一个新的启动项：



C:\Windows\system32>bcdedit /copy {current} /d debug-10101

已将该项成功复制到 {ffdbe9b2-a27a-11ec-b786-fc8640742726}。

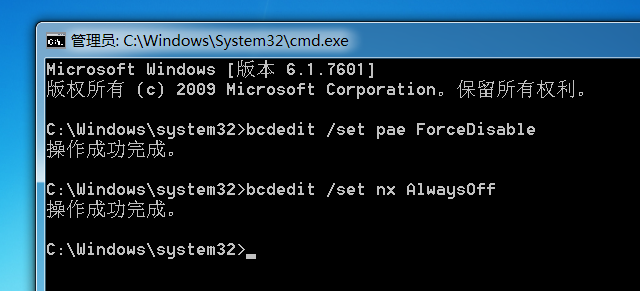
然后重启电脑



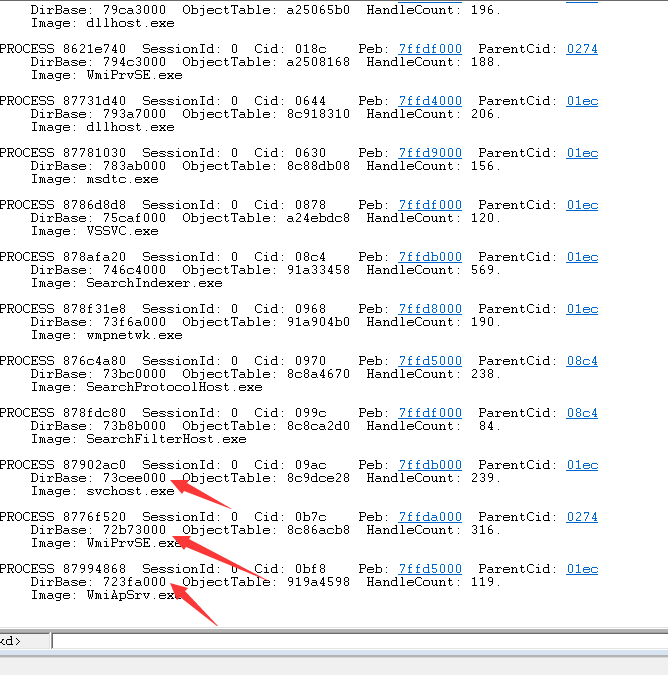
重启后，执行两条命令：

bcdedit /set pae ForceDisable

bcdedit /set nx AlwaysOff



再次重启之后，!process 0 0看看：



这样就设置好了。