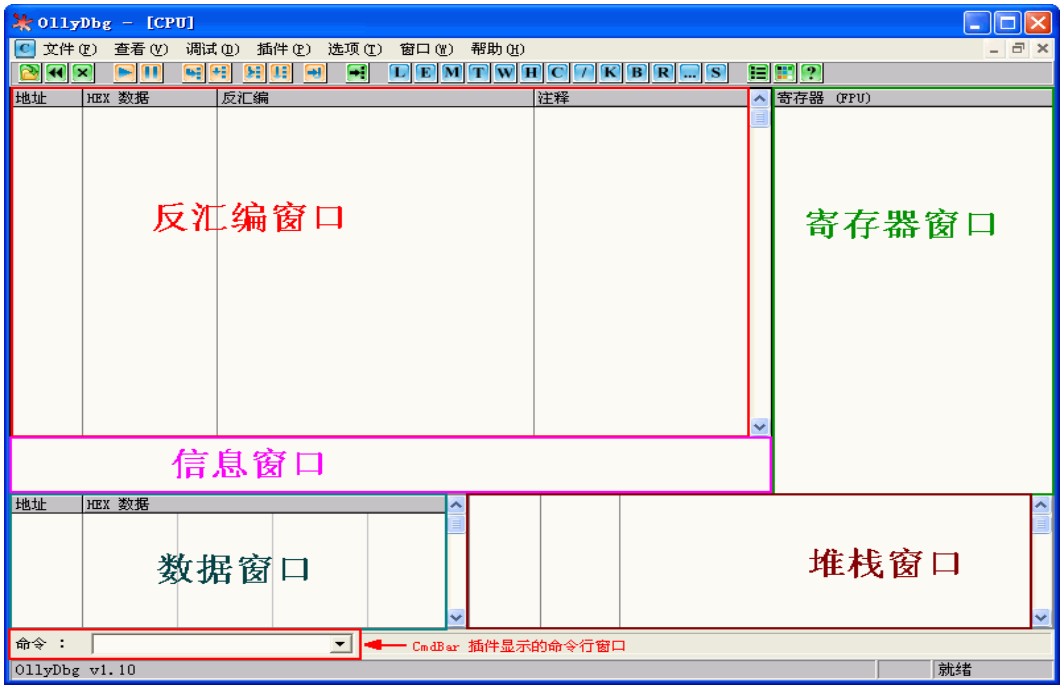
# 学习使用 OllyDbg

## 一、 主要界面




OllyDBG 中各个窗口的功能如上图。简单解释一下各个窗口的功能。

- 反汇编窗口：显示被调试程序的反汇编代码，标题栏上的地址、HEX 数据、反汇编、注释可以通过在窗口中右击出现的菜单 界面选项->隐藏标题 或 显示标题 来进行切换是否显示。用鼠标左键点击注释标签可以切换注释显示的方式。
- 寄存器窗口：显示当前所选线程的 CPU 寄存器内容。同样点击标签寄存器 (FPU) 可以切换显示寄存器的方式。
- 信息窗口：显示反汇编窗口中选中的第一个命令的参数及一些跳转目标地址、字串等。
- 数据窗口：显示内存或文件的内容。右键菜单可用于切换显示方式。
- 堆栈窗口：显示当前线程的堆栈。

## 二、常用指令

- F2：设置断点，只要在光标定位的位置按 F2 键即可，再按一次 F2 键则会删除断点。
- F8：单步步过。每按一次这个键执行一条反汇编窗口中的一条指令，遇

到 CALL 等子程序不进入其代码。

- F7：单步步入。功能同单步步过(F8)类似，区别是遇到 CALL 等子程序时会进入其中，进入后首先会停留在子程序的第一条指令上。
- F4：运行到选定位置。作用就是直接运行到光标所在位置处暂停。
- F9：运行。按下这个键如果没有设置相应断点的话，被调试的程序将直接开始运行。
- CTR+F9：执行到返回。此命令在执行到一个 ret (返回指令)指令时暂停，常用于从系统领空返回到我们调试的程序领空。
- ALT+F9：执行到用户代码。可用于从系统领空快速返回到我们调试的程序领空。

## 三、例子

例如：

```
004013F7  xor     esi, esi
004013F9  push    esi
004013FA  call    00401DA0            //按 F8 键不会进去，而直接路过这个
CALL
004013FF  pop     ecx
00401400  test    eax, eax
```

F7 和 F8 功能键的主要差别就在于若遇到 CALL、LOOP 等指令，F8 键是路过，而 F7 键是跟进去。

```
004013F7  xor     esi, esi
004013F9  push    esi
004013FA  call    00401DA0             //按 F7 键会进入这个 CALL
{
00401DA0  xor     eax, eax          //上面那句 4013FA，按 F7 键就会来到里
00401DA2  push    0
00401DA4  cmp     [esp+8], eax
00401DA8  push    1000
00401DAD  sete    al
……
}
```

OllyDbg 提供了**“Ctrl+F7”**和**“Ctrl+F8”**快捷键，直到用户按 Esc 键、F12 键或遇到其他断点时停止。

当位于某个 CALL 中，这时想返回到调用这个 CALL 的地方时，可以按“**Ctrl+F9**”快捷键执行“执行到返回（Execute till return）”功能。OllyDbg 就会停在遇到的第一个返回命令（RET、RETF 或者 IRET），这样可以很方便地略过一些没用的代码。例如上面的代码，在 401DA0 这行，如果按**“Ctrl+F9”**快捷键就会回到 4013FA 这句。

遇到 RET 指令是暂停还是步过可以在选项里设置，方法是：打开调试设置

选项对话框，在“Trace”页面，设置“After Execting till RET,step over RET（执行到 RET 后，单步步过 RET）”。

如果跟进系统 DLL 提供的 API 函数中，此时想返回到应用程序领空里，可以按快捷键**“Alt＋F9”**执行“Execute till user code（执行到用户代码）”命令。

例如：

```
004013C0  push    ebx
004013C1  push    esi
004013C2  push    edi
004013C3  mov     [ebp-18], esp
004013C6  call    [<&KERNEL32.GetVersion>]   //按 F7 键跟进
KERNEL32.dll 里
004013CC  xor     edx, edx
```

在上面的 4013C6 一行，按 **F7** 键就可跟进系统 KERNEL32.DLL 里的领空：

```
7C8114AB kernel32.GetVersion mov     eax, fs:[18]
7C8114B1                     mov     ecx, [eax+30] //假设当前光标在这行
7C8114B4                     mov     eax, [ecx+B0]
7C8114BA                     movzx   edx, word ptr [ecx+AC]
7C8114C1                     xor     eax, FFFFFFFE
```

像地址 7C8114AB 等都是系统 DLL 所在的地址空间，这时只要按一下快捷键“**Alt＋F9**”就可回到应用程序领空里。代码如下：

```
004013C0   push    ebx
004013C1   push    esi
004013C2   push    edi
004013C3   mov     [ebp-18], esp
004013C6   call    [<&KERNEL32.GetVersion>]
004013CC   xor     edx, edx                    //会返回到此行
```

注意：所谓领空，实际上是指在某一时刻，CPU 的 CS:EIP 所指向的某段代码的所有者。

如果不想单步跟踪，让程序直接运行起来，可以按 F9 键或单击工具栏中的按钮。如果想重新调试目标程序，可以按“**Ctrl+F2**”快捷键或单击工具栏中的按钮，Ollydbg 结束被调试进程并重新加载它。有时程序进入死循环，可以按 F12 键暂停程序。

## 四、设置断点

断点是调试器的一个重要功能，可以让程序中断在需要的地方，从而方便对其分析。最常用的断点是 INT 3 断点，其原理是 Ollydbg 将断点地址处的代码修改为 INT 3 指令。将光标移动到某一行，按 F2 键即可设置一个断点，再按一次 **F2** 键取消断点。也可以用鼠标双击“Hex dump”列中相应的行设置断点，再次双

击取消断点。当关闭程序时，OllyDbg 会自动将当前应用程序的断点位置保存在其安装目录*.udd 文件中，以 便下次运行时，这些断点继续有效。如果将断点设置到当前应用程序代码外，OllyDbg 将会警告。

跟踪函数入口地址：按“**Ctrl+G**”键打开跟随表达式的窗口，输入 GetProcAddress 字符。注意：OllyDbg 里对 API 的大小写敏感，输入的函数名大小写必须正确。单击 OK 按钮后，会来到系统 USER32.DLL 中的 GetDlgItemTextA 函数入口处。按 **F2** 键设个断点，即在 GetProcAddress 函数入口处设了断点（操作系统版本不同，这个函数入口地址是不一样的），如果这个函数被调用，OllyDbg 就会中断。