在使用IDA进行逆向时，经常会碰到需要「定位某个变量被哪些函数访问」或者「某个函数是从什么地方被调用的」。这种跟踪变量或函数的功能在IDA中被称作交叉引用(XREF)，同时IDA还提供了图形生成功能，以更直观的方式显示代码与数据之间的关系。

交叉引用

先总结，IDA中有两类交叉引用：

- 代码交叉引用 CODE XREF
  - 函数调用导致的交叉引用
  - 跳转交叉引用
- 数据交叉引用 DATA XREF
  - 读取交叉引用
  - 写入交叉引用
  - 偏移量交叉引用

所有的交叉引用都是在一个地址引用另一个地址。这些地址可能是代码地址或数据地址。如果引入有向图论，那么就可以把每个地址看成是有向图中的节点，交叉引用看成为边，边的方向就代表着谁引用了谁。

代码交叉引用

代码交叉引用是一个非常重要的概念，因为它可帮助IDA生成控制流图形和函数调用图形，这些图形最直观的用法就是可以帮助我们识别程序的流程，例如判断代码是否被OLLVM的控制流平坦化混淆过。

一个基本的代码交叉引用类似如下：

> .text:00401000 sub_40100 proc near ; CODE XREF: main+2A ↓p

CODE XREF代表这是一个代码交叉引用，后面的地址（这里为_main+2A）是交叉引用的源头地址，即_main+2A处调用了该方法，交叉引用中使用的地址提供了额外的信息，指出交叉引用是在一个名为_main的函数中提出的，具体而言是_main函数中的第0x2A（42）字节。

地址后面总是有一个上行或下行箭头，表示引用位置的相对方向。需要向下滚动才能到达该地址。同样，上行箭头表示引用地址是一个较低的内存地址，需要向上滚动才能到达。

箭头后还包含了一个单字符后缀，代表交叉引用的类型，p代表这是一个调用类型，j则代表这是一个跳转类型的交叉引用。

可能很多人会好奇为什么将例如if else的代码改写成switch case的形式会被成为控制流平坦化，在IDA中，指令转交控制权的方式叫做流（flow）。IDA中有3种基本流：普通流、跳转流和调用流。

普通流（ordinary flow）是一种最简单的流，它表示由一条指令到另一条指令的顺序流。这是所有非分支指令（如ADD）的默认执行流。除了指令在反汇编代码清单中的显示顺序外，正常流没有其他特殊的显示标志。

每个无条件分支指令和条件分支指令将分配到一个跳转流（jump flow）。函数调用则被分配到一个调用流例如call指令。

通过交叉引用我们可以快速的追溯到每个跳转流和调用流的关系流程。

数据交叉引用

数据交叉引用用于跟踪二进制文件访问数据的方式。IDA中最常用的3种数据交叉引用分别用于表示某个位置何时被读取、何时被写入以及何时被引用。下面这张图就展示了这三种数据交叉引用：

```
.data:0040B720 read_it      dd ?                ; DATA XREF: _main+E↑r
.data:0040B720                                  ; _main+25↑r
.data:0040B724 write_it     dd ?                ; DATA XREF: _main+1B↑w
.data:0040B724                                  ; _main+2E↑w ...
.data:0040B728 ref_it       db ? ;              ; DATA XREF: _main+4↑o
.data:0040B729              db ? ;
.data:0040B72A              db ? ;
.data:0040B72B              db ? ;
```

读取交叉引用

读取交叉引用（read cross-reference）表示访问的是某个内存位置的内容，使用r作为单字符后缀，例如上面IDA反汇编中可以得知read_it被main+E处被读取。

同理write_it处的写入交叉引用表示修改write_it处的值，使用w作为单字符后缀，值得注意的是，write_it位置显示的交叉引用以省略号处结束，表明对write_it的交叉引用数量超出了当前的交叉引用显示限制。你可以通过Options▶General对话框中Cross-references选项卡中的Number of displayed xrefs（显示的交叉引用数量）设置修改这个限制。

第三类数据交叉引用为偏移量交叉引用（offset cross-reference），它表示引用的是某个位置的地址（而非内容），使用o作为后缀。通常，代码或数据中的指针操作会导致偏移量交叉引用。例如，数组访问操作一般通过在数组的起始地址上加上一个偏移量来实现。因此，许多全局数组的第一个地址通常可以由偏移量交叉引用来确定。为此，许多字符串数据（在C/C++中，字符串作为字符数组）成为偏移量交叉引用的目标。

交叉引用列表窗口

如前所述，在某个位置显示的交叉引用注释的数量由一个配置控制，其默认设置为2。只要一个位置的交叉引用数量不超出这个限制，你就可以相当直接地访问这些交叉引用。

但如果想要查看某个位置的交叉引用完整列表，第一种方法是打开与某一特定位置有关的交叉引用子窗口。将光标放在一个或多个交叉引用的目标地址上，并选择View▶OpenSubviews▶Cross-References（查看▶打开子窗口▶交叉引用），即可打开指定位置的交叉引用完整列表。

第二种访问交叉引用列表的方法是选中一个你感兴趣的符号名称，在菜单中选择Jump▶Jump to xref（使用热键CTRL+X）打开一个对话框，其中列出了引用选中符号的每个位置。

交叉引用列表可用于迅速确定调用某个特殊函数的位置。例如如果想要确定程序中哪些地方调用了C标准库中的strcpy函数，可以用上述方法打开该函数的交叉引用列表窗口，甚至可以添加一段包含strcpy文本的注释，并使用该注释激活“交叉引用”对话框，因为IDA会讲注释中的符号名称也作为一个操作数处理，通过点击注释中的strcpy也能快速打开交叉引用列表。