本章的目的是厘清IDC脚本如何执行、如何编写以及IDC脚本的语法。

为了满足用户对IDA的定制需求，IDA集成了一个脚本引擎，让用户从编程角度对IDA的操作进行全面控制。

通过IDA脚本可以从自动化和分析两个角度控制IDA：

- 自动化：执行一系列动作和操作步骤
- 分析：从分析角度IDA脚本语言可以看成是一种查询语言，能够访问IDA数据库的内容。

IDC和IDAPython

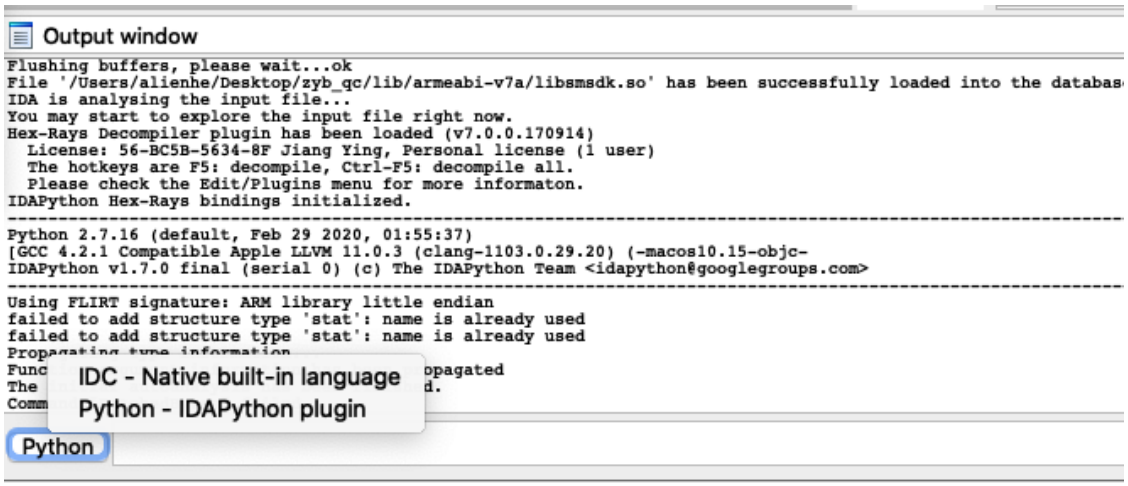
IDA使用两种不同的语言编写脚本，原始嵌入式脚本语言叫做IDC，因为与C语法非常相似。

其次，还可以通过集成IDA Python插件来通过Python编写脚本。

执行脚本的常用方法

三个菜单选项 和 一个命令行

1. File -> Script File 执行一个独立的IDC文件
2. File -> IDC Command 执行少数几个IDC语句
3. File -> Python Command 执行少数Python语句
4. IDA工作区最下方输出窗口下面有一个命令行输入框



IDC

数据类型

学习一门编程语言，首先学习基本的数据类型，满足基本的变量声明和编程语句的编写。

IDC是一种类型松散的语言，IDC使用3种数据类型：整数（IDA文档使用类型名称long）、字符串和浮点值，其中绝大部分的操作针对的是整数和字符串。字符串被视为IDC中的本地数据类型，因此，你不需要跟踪存储一个字符串所需的空间，或者一个字符串是否使用零终止符。从IDA5.6开始，IDC加入了许多变量类型，包括对象、引用和函数指针。

变量声明

局部变量声明：auto 全局变量声明：extern

与大多数语言声明变量类似，除了：

- 关键字有所不同
- 可以在任何函数定义的内部和外部声明全局变量，但不能在声明全局变量时为其提供初始值

```
extern outsideGlobal;

static main(){

    extern insideGlobal;
    outsideGlobal = "Global";
    insideGlobal = 1;
    auto local = 2;
}
```

表达式

支持：

- 基本运算符
- 三元运算符
- 逗号运算符
- 分片运算符

不支持：

- op=类型的复合赋值运算符例如+= 、 >>= 、 *=

由于字符串是IDC中的本地类型，因此，IDC中的一些字符串运算与C中的字符串运算有所不同。在IDC中不需要使用strcpy或strdup、strcat等函数，类似Java，“Hello”+“World”将得到“HelloWorld”。

分片运算符使用方括号和起始索引（包括）与结束索引（不包括）来指定（至少需要一个索引）字符串中一个与数组类似的变量的子序列，虽然IDC中并没有数组数据类型，但你可以使用分片运算符来处理IDC字符串，就好像它们是数组一样。

IDC语句

IDC中每条语句均以分号结尾，和C基本类似，且支持try/catch和throw语句，但不支持switch-case结构。

IDC不存在块作用域，即花括号内声明的变量可以在括号外引用，但是无法在函数外引用。

IDC函数

IDC脚本中支持函数声明，但是IDC命令对话框中不允许函数声明。

IDC中使用static关键字引入一个用户定义的函数，参数列表为逗号分割的参数名列表。例如

```
static my_func(x,y,z){

}
```

IDA5.6之前，参数传递仅有值传递的方式，之后则引入了按地址传递，两者传递方式由调用函数的方式决定，在函数调用时使用&表明该函数采用地址传递方式传递参数。

返回值上与C一致，使用return关键字，并且不显示返回时将默认返回0。但是可通过函数的不同执行路径返回不同的数据类型。换言之，某些情况下，一个函数返回一个字符串；而在其他情况下，这个函数却返回一个整数。

IDA5.6之后，支持将函数当作对象作为参数和返回值进行传递。

IDC程序文件中要求至少定义一个没有参数的main函数，另外，主程序文件中还必须包含idc.idc文件用以获得idc的一些宏定义

```
#include <idc.idc>
static main(){
    // some code
}
```

IDC对象

IDC定义了一个称为object的根类，是所有类的父类。但是IDC并不使用访问权限修饰符即private\public等，所有类成员均为公共类，类声明仅包含类成员函数的定义。要在类中创建数据成员，你只需要创建一个给数据成员赋值的赋值语句即可。例如：

```
// 声明类
class DemoClass{
        DemoClass(x,y){
            this.a=x;// 声明成员变量
            this.b=y;
        }
        ~DemoClass(){}//析构函数

}
```

预处理指令

IDC支持与C相同的预处理指令：

- #inlcude
- #define
- #ifdef
- #else
- #endif
- #undef

关联IDC脚本与热键

如果期望将你的脚本与某个快捷键绑定，可以应用到IDC启动时默认执行的ida.idc脚本。

每次启动IDA，它都会执行<IDADIR>/idc/ida.idc中的脚本。这个脚本的默认版本包含一个空的main函数，因此，它不执行任何操作。为了将热键与脚本关联起来，你需要在ida.idc文件中添加两行代码。在第一行代码中，必须添加一个include指令，将脚本文件包含在ida.idc文件中。在第二行代码中，必须在main函数中添加一个对AddHotkey函数的调用，将特定的热键与IDC脚本关联起来。

```
#include <idc.idc>
#include <my_script>

static main(){
    AddHotKey("z","MyFunc");//按z键时将会触发MyFunc函数的调用
}
```