x64逆向：异常处理

在x64下，函数起始不在注册SEH，而异常相关信息全局存储在`.pdata`节中

所以，同x86的区别**只有**寻找`FuncInfo`结构的路径不同：

- 要从`.pdata`节中得到`RUNTIME_FUNCTION`
- 通过`RUNTIME_FUNCTION`的第三个成员找到`UNWIND_INFO_HDR`
- 通过`UNWIND_INFO_HDR`标识的`UNWIND_CODE`结构的个数，找到后续的`UNWIND_CODE`
- 最后由`UNWIND_INFO_HDR`的`Ver3_Flags`成员来判断`UNWIND_CODE`后续的异常处理函数和`FuncInfo`

相关结构

RUNTIME_FUNCTION

```
RUNTIME_FUNCTION struc ; (sizeof=0xC, mappedto_5)
    FunctionStart   dd ?                    ; offset rva
    FunctionEnd     dd ?                    ; offset rva pastend
    UnwindInfo      dd ?                    ; offset rva
RUNTIME_FUNCTION ends
```

- `FunctionStart`：函数起始地址（偏移）
- `FunctionEnd`：函数结束地址（偏移）
- `UnwindInfo`：展开信息`UNWIND_INFO`的地址（偏移）

UNWIND_INFO_HDR

```
UNWIND_INFO_HDR struc ; (sizeof=0x4, mappedto_6)
    Ver3_Flags      db ?                    ; base 16
    PrologSize      db ?                    ; base 16
    CntUnwindCodes  db ?                    ; base 16
    FrReg_FrRegOff  db ?                    ; base 16
UNWIND_INFO_HDR ends
```

- `Ver3_Flags`：低3位Version，高5位Flags
    - Flags标识存在`UNW_LFAG_EHANDLER(1)`，则`UNWIND_CODE`结构后会跟随一个函数地址和一个`FuncInfo`
- `PrologSize`：序言大小
- `CntUnwindCodes`：展开代码数组个数，标识后续跟着多少个`UNWIND_CODE`结构
- `FrReg_FrRegOff`：低4位为帧寄存器，高4位为帧寄存器偏移量

UNWIND_CODE

```
UNWIND_CODE     struc ; (sizeof=0x2, mappedto_7)
    PrologOff       db ?                    ; base 16
    OpCode_OpInfo   db ?                    ; base 16
UNWIND_CODE     ends
```

- `PrologOff`：序言中的偏移
- `OpCode_OpInfo`：低4位为展开操作代码，高4位为操作信息

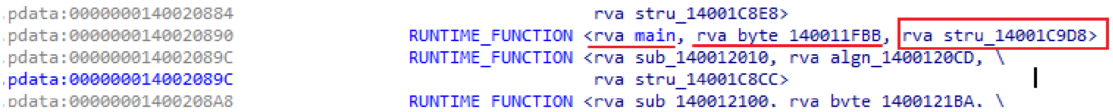
实例

```
int main(void)
{
    try {
        throw 1;
    } catch (int e) {
        printf("catch int: %d\n", e);
    } catch (float e) {
        printf("catch float: %f\n", e);
    } catch (double e) {
        printf("catch double: %lf\n", e);
    } catch (long long e) {
        printf("catch long long: %llu\n", e);
    } catch (...) {
        printf("catch ...\n");
    }

    system("pause");
    return 0;
}
```

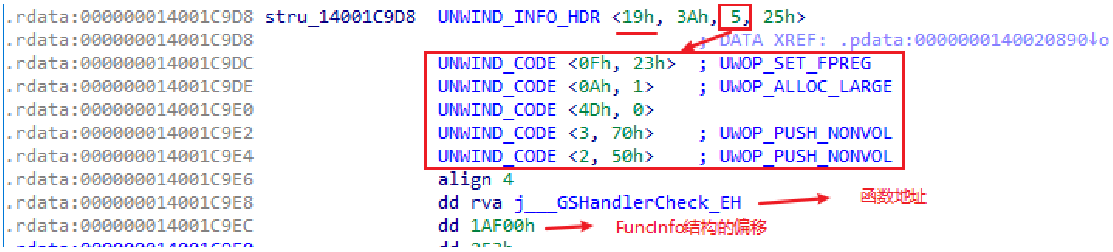
从.pdata定位

在`.pdata`中，`RUNTIME_FUNCTION`结构表中记录了main函数中有异常信息



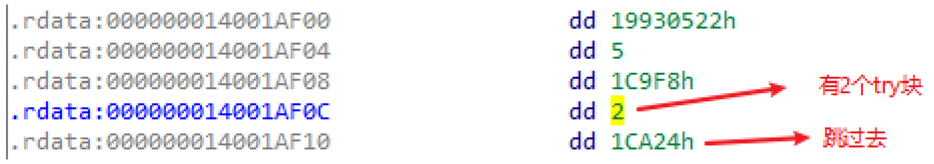
```
.pdata:0000000140020884                          rva stru_14001C8E8>
.pdata:0000000140020890           RUNTIME_FUNCTION <rva main, rva byte_140011FBB, rva stru_14001C9D8>
.pdata:000000014002089C           RUNTIME_FUNCTION <rva sub_140012010, rva algn_1400120CD, \
.pdata:000000014002089C                          rva stru_14001C8CC>
.pdata:00000001400208A8           RUNTIME_FUNCTION <rva sub_140012100, rva byte_1400121BA, \
```

通过第三个成员转到`UNWIND_INFO_HDR`



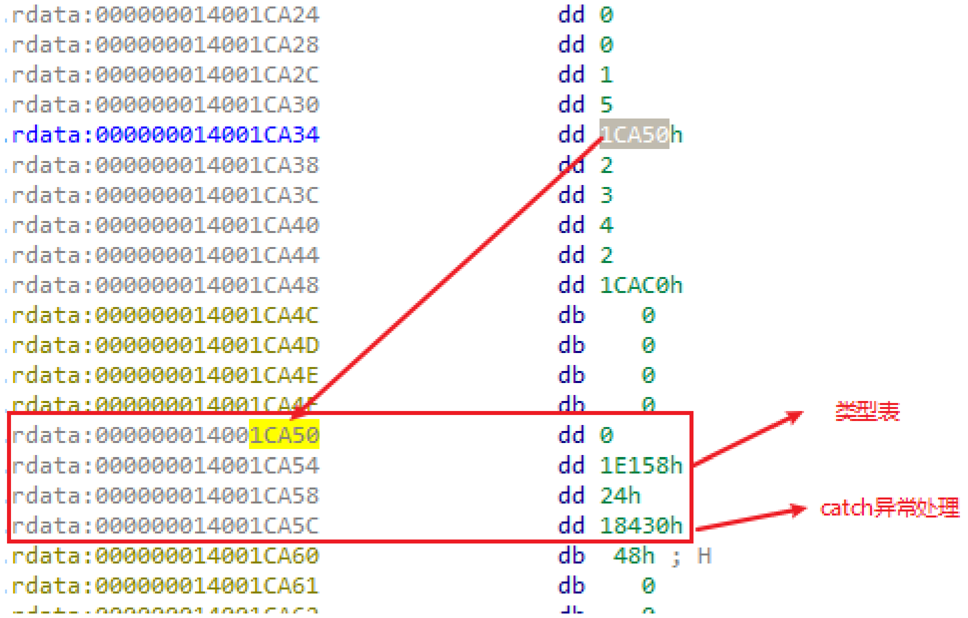
```
.rdata:000000014001C9D8 stru_14001C9D8  UNWIND_INFO_HDR <19h, 3Ah, 5, 25h>
.rdata:000000014001C9D8                                 ; DATA XREF: .pdata:0000000140020890↓o
.rdata:000000014001C9DC                 UNWIND_CODE <0Fh, 23h>  ; UWOP_SET_FPREG
.rdata:000000014001C9DE                 UNWIND_CODE <0Ah, 1>    ; UWOP_ALLOC_LARGE
.rdata:000000014001C9E0                 UNWIND_CODE <4Dh, 0>
.rdata:000000014001C9E2                 UNWIND_CODE <3, 70h>    ; UWOP_PUSH_NONVOL
.rdata:000000014001C9E4                 UNWIND_CODE <2, 50h>    ; UWOP_PUSH_NONVOL
.rdata:000000014001C9E6                 align 4
.rdata:000000014001C9E8                 dd rva j___GSHandlerCheck_EH
.rdata:000000014001C9EC                 dd 1AF00h
```

此后跟x86一样了



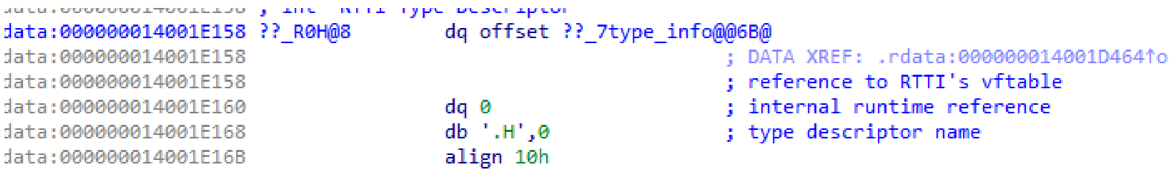
```
.rdata:000000014001AF00                 dd 19930522h
.rdata:000000014001AF04                 dd 5
.rdata:000000014001AF08                 dd 1C9F8h
.rdata:000000014001AF0C                 dd 2
.rdata:000000014001AF10                 dd 1CA24h
```

跳转到VA = Base + 0x1ca24

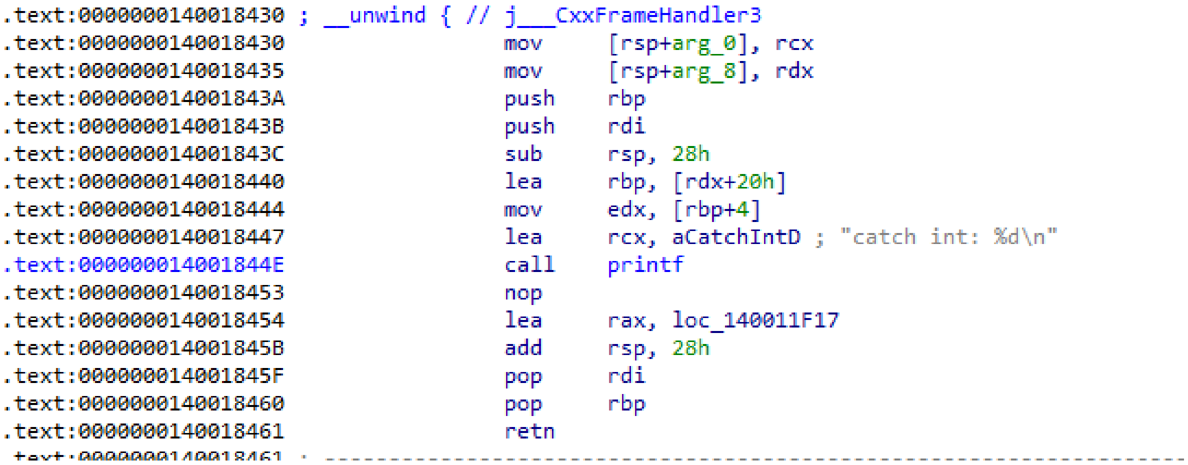


```
.rdata:000000014001CA24                dd 0
.rdata:000000014001CA28                dd 0
.rdata:000000014001CA2C                dd 1
.rdata:000000014001CA30                dd 5
.rdata:000000014001CA34                dd 1CA50h
.rdata:000000014001CA38                dd 2
.rdata:000000014001CA3C                dd 3
.rdata:000000014001CA40                dd 4
.rdata:000000014001CA44                dd 2
.rdata:000000014001CA48                dd 1CAC0h
.rdata:000000014001CA4C                db    0
.rdata:000000014001CA4D                db    0
.rdata:000000014001CA4E                db    0
.rdata:000000014001CA4F                db    0
.rdata:000000014001CA50                dd 0
.rdata:000000014001CA54                dd 1E158h
.rdata:000000014001CA58                dd 24h
.rdata:000000014001CA5C                dd 18430h
.rdata:000000014001CA60                db  48h ; H
.rdata:000000014001CA61                db    0
.rdata:000000014001CA62                db    0
```

就可得到类型和catch块



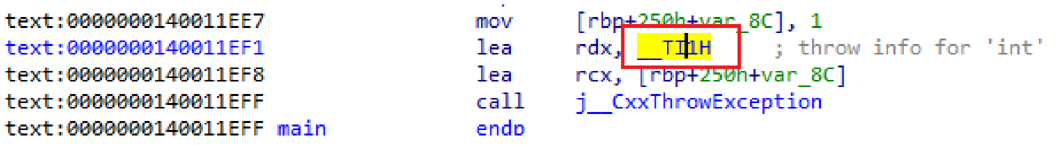
```
data:000000014001E158 ??_R0H@8        dq offset ??_7type_info@@6B@
data:000000014001E158                                          ; DATA XREF: .rdata:000000014001D464↑o
data:000000014001E158                                          ; reference to RTTI's vftable
data:000000014001E160                 dq 0                     ; internal runtime reference
data:000000014001E168                 db '.H',0                ; type descriptor name
data:000000014001E16B                 align 10h
```



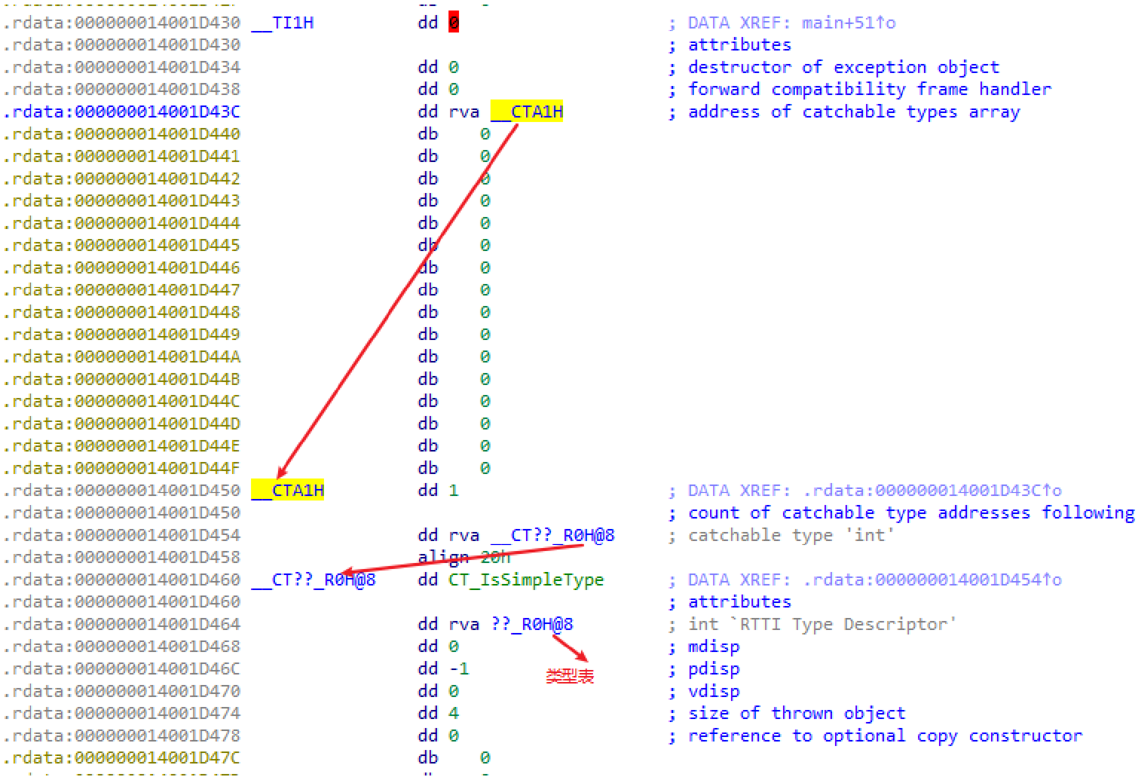
```
.text:0000000140018430 ; __unwind { // j___CxxFrameHandler3
.text:0000000140018430                 mov     [rsp+arg_0], rcx
.text:0000000140018435                 mov     [rsp+arg_8], rdx
.text:000000014001843A                 push    rbp
.text:000000014001843B                 push    rdi
.text:000000014001843C                 sub     rsp, 28h
.text:0000000140018440                 lea     rbp, [rdx+20h]
.text:0000000140018444                 mov     edx, [rbp+4]
.text:0000000140018447                 lea     rcx, aCatchIntD ; "catch int: %d\n"
.text:000000014001844E                 call    printf
.text:0000000140018453                 nop
.text:0000000140018454                 lea     rax, loc_140011F17
.text:000000014001845B                 add     rsp, 28h
.text:000000014001845F                 pop     rdi
.text:0000000140018460                 pop     rbp
.text:0000000140018461                 retn
.text:0000000140018461 ; ---------------------------------------------------------------------------
```

从throw定位

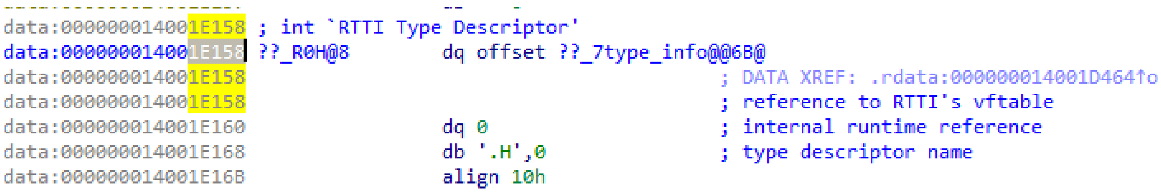
从throw的第二个参数去找



```
text:0000000140011EE7                 mov     [rbp+250h+var_8C], 1
text:0000000140011EF1                 lea     rdx, __TI1H      ; throw info for 'int'
text:0000000140011EF8                 lea     rcx, [rbp+250h+var_8C]
text:0000000140011EFF                 call    j__CxxThrowException
text:0000000140011EFF main            endp
```

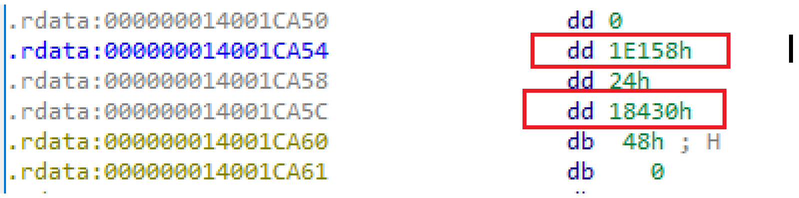


```
.rdata:000000014001D430 __TI1H          dd 0                    ; DATA XREF: main+51↑o
.rdata:000000014001D430                                         ; attributes
.rdata:000000014001D434                 dd 0                    ; destructor of exception object
.rdata:000000014001D438                 dd 0                    ; forward compatibility frame handler
.rdata:000000014001D43C                 dd rva __CTA1H          ; address of catchable types array
.rdata:000000014001D440                 db    0
.rdata:000000014001D441                 db    0
.rdata:000000014001D442                 db    0
.rdata:000000014001D443                 db    0
.rdata:000000014001D444                 db    0
.rdata:000000014001D445                 db    0
.rdata:000000014001D446                 db    0
.rdata:000000014001D447                 db    0
.rdata:000000014001D448                 db    0
.rdata:000000014001D449                 db    0
.rdata:000000014001D44A                 db    0
.rdata:000000014001D44B                 db    0
.rdata:000000014001D44C                 db    0
.rdata:000000014001D44D                 db    0
.rdata:000000014001D44E                 db    0
.rdata:000000014001D44F                 db    0
.rdata:000000014001D450 __CTA1H         dd 1                    ; DATA XREF: .rdata:000000014001D43C↑o
.rdata:000000014001D450                                         ; count of catchable type addresses following
.rdata:000000014001D454                 dd rva __CT??_R0H@8     ; catchable type 'int'
.rdata:000000014001D458                 align 20h
.rdata:000000014001D460 __CT??_R0H@8    dd CT_IsSimpleType      ; DATA XREF: .rdata:000000014001D454↑o
.rdata:000000014001D460                                         ; attributes
.rdata:000000014001D464                 dd rva ??_R0H@8         ; int `RTTI Type Descriptor'
.rdata:000000014001D468                 dd 0                    ; mdisp
.rdata:000000014001D46C                 dd -1                   ; pdisp
.rdata:000000014001D470                 dd 0                    ; vdisp
.rdata:000000014001D474                 dd 4                    ; size of thrown object
.rdata:000000014001D478                 dd 0                    ; reference to optional copy constructor
.rdata:000000014001D47C                 db    0
```



```
data:000000014001E158 ; int `RTTI Type Descriptor'
data:000000014001E158 ??_R0H@8        dq offset ??_7type_info@@6B@
data:000000014001E158                                         ; DATA XREF: .rdata:000000014001D464↑o
data:000000014001E158                                         ; reference to RTTI's vftable
data:000000014001E160                 dq 0                    ; internal runtime reference
data:000000014001E168                 db '.H',0               ; type descriptor name
data:000000014001E16B                 align 10h
```

搜索类型表的二进制偏移量，可得到哪里引用了此地址（快捷键ALT + T）



```
.rdata:000000014001CA50                 dd 0
.rdata:000000014001CA54                 dd 1E158h
.rdata:000000014001CA58                 dd 24h
.rdata:000000014001CA5C                 dd 18430h
.rdata:000000014001CA60                 db  48h ; H
.rdata:000000014001CA61                 db    0
```