IDAPython 6.8

long char void bool iterator

См. файл IDA_DIR\python\idc.py

Некоторые константы

```
BADADDR = 0xffffffffL
BADSEL = 0xffffffffL
MAXADDR = 0xFF000000
SIZE_MAX = 0xffffffffL
```

Взаимодействие с пользователем

```
Message(format, ...)      AskStr(default, prompt)
Warning(format, ...)      AskFile(doSave, mask, prompt)
ChooseFunction(title)     AskYN(default, prompt)
```

Диалоговые окна

```
Jump(ea)
ScreenEA()
here()
```

Текущая позиция

Debugger Hooks

```
AddBpt(ea) / DelBpt(ea) / EnableBpt(ea)
GetBptQty() / GetBptEA(n)
SetRegValue(val, reg) / GetRegValue(reg)
```

Анализ

```
AnalyseArea(sEA, eEA) / AutoMark(ea, QType)
DeleteAll() / AutoMark2(start, end, QType)
Demangle(name, disableMask)
```

Точки входа

```
AddEntryPoint(ord, ea, name, makeCode)
RenameEntryPoint(ord, name) / GetEntryPoint(ord)
GetEntryOrdinal(index) / GetEntryPointQty()
```

Сегменты

```
Segments()                Итерация по сегментам
FirstSeg() / NextSeg(ea)
SegName(ea) / SegByName(name)
SegStart(ea) / SegEnd(ea)
```

Поиск по БД

```
FindBinary(ea, flags, binary)
FindText(ea, flags, row, col, text)
FindImmediate(ea, flag, value)
FindCode(ea, flags) / FindFuncEnd(ea)
FindVoid(ea, flag) / FindData(ea, flags)
```

Перекрёстные ссылки

```
CodeRefsTo(ea, flow)      Ссылки кода на/из адрес(а)
CodeRefsFrom(ea, flow)    flow  Следовать потоку кода?
DataRefsTo(ea)            Ссылки данных на/из адрес(а)
DataRefsFrom(ea)
XrefsFrom(ea, flags=0)    Все ссылки на/из адрес(а)
XrefsTo(ea, flags=0)
```

id – уникальный идентификатор, idx – индекс,
rpt – повторяемый ли комментарий,
mOff – смещение элемента структуры

Структуры

Структура

```
AddStrucEx(idx, name, isUnion) / DelStruc(id) / IsUnion(structId)
GetStrucQty() / GetStrucId(idx) / GetStrucIdByName(name) / GetStrucSize(id)
GetStrucIdx(id) / GetStrucName(id) / GetStrucComment(id, rpt)
```

Поля стуктуры

```
AddStrucMember(id, name, mOff, flag, typeid, nbytes)
SetMemberName(id, mOff, name) / SetMemberComment(id, mOff, cmnt, rpt)
GetMemberQty(id) / GetMemberName(id, mOff) / GetMemberComment(id, mOff, rpt)
GetMemberSize(id, mOff) / GetMemberStrId(id, mOff) / DelStrucMember(id, mOff)
```

Перечисления (Enums)

```
GetEnumQty()              GetnEnum(idx)
GetEnumIdx(id)            GetEnum(name)
GetEnumName(id) / GetEnumSize(id)
GetEnumCmt(id, rpt)
GetConstByName(name) / GetConstValue(id)
GetConstName(id) / GetConstCmt(id, rpt)
```

Функции

```
LocByName(name)           Линейный адрес метки с именем name
Functions(startEA, endEA) Итерация по функциям
NextFunction(ea) / PrevFunction(ea)
GetFunctionName(ea)       Имя функции или пустая строка
GetFuncOffset(ea)         ИмяФункции+СмещениеОтНачалаФункции
MakeFunction(begin, end)  Управление функцией
DelFunction(ea)
SetFunctionEnd(ea, end)
SetFunctionCmt(ea, cmt, rpt)   Комментарий к функции
GetFunctionCmt(ea, rpt)        rpt 0 – неповторяемый 1 - повторяемый
```

Управление атрибутами функции

```
GetFunctionFlags(ea)
SetFunctionFlags(ea, flags)
```

FUNC_NORET	FUNC_FAR
FUNC_LIB	FUNC_STATIC
FUNC_FRAME	FUNC_USERFAR
FUNC_HIDDEN	

Взаимодействие со стековым регистром (SP)

```
GetSpd(ea) / GetSpDiff(ea) / SetSpDiff(ea, delta)
```

Взаимодействие с фреймом функции

```
GetFrame(ea) / MakeFrame(ea, nVars, nRegs, nArgs)
GetFrameSize(ea)          Весь фрейм
GetFrameRegsSize(ea)      Регистр во фрейме
GetFrameLvarSize(ea)      Переменные
GetFrameArgsSize(ea)      Аргументы
```

Размер в байтах

Чтение/запись данных

```
*Byte(ea)                 Patch*Byte(ea, value)
*Word(ea)                 PatchWord(ea, value)
*Dword(ea)                PatchDword(ea, value)
isLoaded(ea)              Актуальный адрес? * префикс Dbg
```

Инструкции

```
FuncItems(start)          Адреса инструкций функции
ItemHead(ea)              Адрес начала инструкции/данных
ItemEnd(ea)               Адрес после инструкции/данных
ItemSize(ea)              Размер элемента от ea до конца
```

Цвет фона

```
GetColor(ea, what)
SetColor(ea, what, color)
```

what

```
CIC_ITEM (1)
CIC_FUNC (2)
CIC_SEGM (3)
```

RGB (hex 0xBBGGRR)

Листинг, комментарии, операнды

```
MakeComm(ea, cmnt)
MakeRptCmt(ea, cmnt)      Комментарии  type 0 - обычный 1 - повторяемый
CommentEx(ea, type)
ExtLin*(ea, n, line)
DelExtLn*(ea, n)          Многострочный текст в коде
Line*(ea, n)              * – это A / B   n – номер строки
```

```
MakeCode(ea)
MakeByte(ea)              Word Dword
MakeUnkn(ea, flags)       Интерпретация кода / данных
```

```
DOUNK_SIMPLE    0
DOUNK_EXPAND    1
DOUNK_DELNAMES  2
```

Строки

```
MakeStr(begin, end)
GetString(ea, len=-1, sType=0)
```

```
ASCSTR_TERMCHR 0   ASCSTR_C      0
ASCSTR_PASCAL  1   ASCSTR_LEN2   2
ASCSTR_UNICODE 3   ASCSTR_LEN4   4
ASCSTR_ULEN2   5   ASCSTR_ULEN4  6
ASCSTR_LAST    6
```

```
MakeName(ea, name)        Имя / метка кода / данных
Name(ea)
```

Работа с листингом

```
GetDisasm(ea)             mov ebp, 100h
GetMnem(ea)               mov
GetOpnd(ea, n)            0) ebp 1) 100h
GetOperandValue(ea, n)    0) 0   1) 256
GetOpType(ea, n)          Зависит от архитектуры
```

Интерпретация операндов

```
OpBinary(ea, n)           OpOctal(ea, n)
OpDecimal(ea, n)          OpHex(ea, n)
OpChr(ea, n)              OpSign(ea, n)
OpSeg(ea, n)              OpStkvar(ea, n)
OpOff(ea, n, base)        / OpOffEx /
OpEnumEx(ea, n, enumid, serial)
OpStroffEx(ea, n, structId, delta)
OpAlt(ea, n, val)         Пользовательский операнд-строка
AltOp(ea, n)
```

n Номер (с нуля). -1 - все base Базовый адрес в байтах

Pavel Rusanov v.1.0

IDAPython 7.x

Python 2.7

long char void bool iterator

Доп.информацию см. в модулях `IDA_DIR\python\`

Обратная совместимость: `IDA_DIR\python\idc_bc695.py`

Константы

```
BADADDR = BADSEL = 0xffffffffL
MAXADDR = 0xFF000000
SIZE_MAX = 0xffffffffL
```

Анализ данных

```
plan_and_wait(start, end)          # Анализировать диапазон
auto_mark_range(start, end, QType)
delete_all_segments()
demangle_name(name, disableMask)
```

Точки входа

```
*Entries()
add_entry(ord, ea, name, makeCode)
rename_entry(ord, name) | get_entry(ord)
get_entry_ordinal(index) | get_entry_qty()
```

Ссылки (XRef)

```
= Ссылки кода = на / из адреса
*CodeRefsTo(ea, flow)
*CodeRefsFrom(ea, flow)
= Ссылки данных = на / из адреса
*DataRefsTo(ea)
*DataRefsFrom(ea)
= Все ссылки = на / из адреса
*XrefsTo(ea, flags=0)
*XrefsFrom(ea, flags=0)
```

Следовать нормальному *потоку* кода?

```
ida_xref.XREF_*
USER TAIL BASE MASK PASTEND ALL FAR DATA
```

Функции

```
Обход функций
*Functions(ea, end)
get_[prev|next]_func(ea)
get_name_ea_simple(name)   # Линейный адрес метки name
get_func_name(ea)          # Имя функции или пустая строка
get_func_off_str(ea)       # Строка 'funcname+offset'
= Работа с функцией =
set_func_end(ea, end)
add_func(ea, end=BADADDR) | del_func(ea)
= Комментарий функции =
get_func_cmt(ea, rpt)
set_func_cmt(ea, cmt, rpt)
```

Неповторяемый (0) Повторяемый (1)

```
= Атрибуты функции =
get_func_attr(ea, attr)
set_func_attr(ea, attr, val)
```

```
FUNCATTR_*
START END FLAGS FRAME
FRSIZE FRREGS REFQTY FPD
COLOR OWNER ARGSIZE
```

```
= Указатель стека функции =
get_sp_delta(ea)
add_user_stkpnt(ea, delta) | get_spd(ea)
= Кадр стека функции =
set_frame_size(ea, lvsize, frregs, args)
```

Локальные перем-е | Сохранённые регистры | Аргу-менты

```
get_frame_size(ea)          # Весь кадр
get_frame_regs_size(ea)     # Регистры
get_frame_lvar_size(ea)     # Локальные перем-е
get_frame_args_size(ea)     # Аргументы
get_frame_id(ea)            # ID структуры кадра стека
```

Размер (в байтах)

* idautils

Высоко-уровневый модуль

Ввод пользователя

= Получить / установить позицию в коде =

```
get_screen_ea() | here() | jumpto(ea)
```

= GUI-диалоги =

```
msg(msg) | warning(msg)
ask_yn(dflt, prompt) | choose_func(title)
idaapi.ask_str(dflt, history, prompt)
idaapi.ask_file(doSave, mask, prompt)
```

Поиск в БД

```
find_binary(ea, flag, bin, rdx=16)
find_text(ea, flag, row, col, text)
find_imm(ea, flag, value)
find_[code|data|suspop](ea, flag)
find_func_end(ea)
```

```
SEARCH_* UP DOWN
NEXT CASE REGEX
NOBRK NOSHOW
```

Элементы кода (Items)

```
next|prev_addr(ea) | next|prev_not_tail(ea)
get_item_[head|end](ea)        # Адрес начала/конца элемента
get_item_size(ea)              # Размер от ea до конца элемента
next|prev_head(ea, maxea=BADADDR, minea=0)
*FuncItems(ea)                 # Адреса элементов функции
*Heads(ea=None, end=None)      # Адреса элементов в диапазоне [ea; end)
```

item – это инструкция или данные

idc idaapi

Модули для большинства функций

Сегменты

```
get_segm_attr(ea, attr) | get_segm_[start|end](ea)
set_segm_attr(ea, attr, value)
```

```
SEGATTR_*
START , END , ORGBASE , ALIGN , COMB , PERM , BITNESS ,
FLAGS , SEL , ES , CS , SS , DS , FS , GS , TYPE , COLOR
```

```
get_segm_name(ea) | selector_by_name(name)
= Обход сегментов =
*Segments()
get_first_seg() | get_next_seg(ea)
```

Чтение / Запись в БД

```
get_wide_[byte|word|dword](ea) | patch_[byte|word|dword|qword](ea, val)
get_qword(ea)
get_bytes(ea, size, useDbg=False)
is_loaded(ea)                  # Байт инициализирован?
```

Цвет фона

```
get_color(ea, what)
set_color(ea, what, color)
```

```
CIC_*
*ITEM (1)
*FUNC (2)
*SEGM (3)
```

```
RGB
hex
0xBBGGRR
```

Отладка (debugger)

```
add_bpt(ea, size=0, type=BPT_DEFAULT) | del_bpt(ea)
enable_bpt(ea, flag)
get_bpt_qty() | get_bpt_ea(n)
```

```
BPT_*
WRITE (1) RDWR (3) SOFT (4)
EXEC (8) DEFAULT (12)
```

```
get_reg_value(reg) | read_dbg_memory(ea, size)
set_reg_value(val, reg) | write_dbg_memory(ea, data)
step_[into|over]() | run_to(ea) | read_dbg_[byte|word|dword|qword](ea)
step_until_ret()
start_process(path, args, sdir)
```

Перечисления (enums)

```
get_enum_qty() | getn_enum(idx)
get_enum_[idx|size](id)
get_enum_member_by_name(name)
get_enum_member_value(id)
get_enum_[name|member_name](id)
get_enum_[cmt|member_cmt](id, rpt)
```

id – Уникальный идентификатор
idx – индекс
rpt – повторяемый комментарий?
mOff – смещение поля структуры

Структуры (structs)

```
= Структура = add_struc(idx, name, isUnion)
*Structs() | del_struc(id) | is_union(id)
get_struc_qty() | get_struc_by_idx(idx)
get_struc_id(name) | get_struc_name(id)
get_struc_[idx|size](id) | get_struc_cmt(id, rpt)
```

```
= Поля структуры = get_member_qty(id)
get_member_[size|strid](id, mOff)
get_member_name(id, mOff)
get_member_cmt(id, mOff, rpt)
set_member_name(id, mOff, name)
set_member_cmt(id, mOff, cmnt, rpt)
add_struc_member(id, name, mOff, flag, type, nbytes, tgt=-1, tdelta=0, rtype=REF_OFF32)
del_struc_member(id, mOff)
```

FF_* константы см. ниже | *Зависит от flag* | *Размер поля* | *Целевой адрес* | *Разница смещений* | *REF_* константы см. ниже*

Листинг, комментарии, операнды

```
= Комментарии =
get_cmt(ea, rpt)
set_cmt(ea, cmnt, rpt)   rpt: 0 - usual, 1 - repeatable
```

Многострочный комментарий == в коде ==

```
get_extra_cmt(ea, n)     # Line number
del_extra_cmt(ea, n)
update_extra_cmt(ea, n, line)
```

```
= Отображение кода / данных =
create_insn(ea)
create_[byte|word|dword|qword](ea)
del_items(ea, flags=0, size=1)
make_array(ea, n)        # Массив из n эл-тов того же типа, как эл-т по адресу ea
create_data(ea, flags, size, tid)
```

```
DELIT_* SIMPLE (0)
EXPAND (1) DELNAMES (2)
```

```
FF_* BYTE WORD DWORD QWORD TBYTE OWORD
STRLIT STRUCT FLOAT DOUBLE PACKREAL ALIGN
```

ID структуры

```
= Имя (метка) кода / данных =
*Names()
set_name(ea, name, sn_flags=SN_CHECK)
get_name(ea, gtn_flags=0)
```

```
GN_* VISIBLE
COLORED DEMANGLED
STRICT SHORT LONG
LOCAL ISRET NOT_ISRET
```

```
SN_* LOCAL CHECK NOCHECK
PUBLIC NON_PUBLIC WEAK
NOLIST AUTO NON_WEAK
NOWARN NON_AUTO
```

```
= Строки =
*Strings() | get_str_type(ea)
get_strlit_contents(ea, len=-1, sType=0)
ida_bytes.create_strlit(ea, len, sType)
```

```
STRTYPE_*
TERMCHR (0) C (0) C16 (1) C_32 (2) PASCAL (64) PASCAL_16 (65)
LEN2 (128) LEN2_16 (129) LEN4 (192) LEN4_16 (193)
```

Создать строку с глобально заданным типом

```
create_strlit(ea, end)
```

end=BADADDR для авто-определения конца строки

```
= Взаимодействие с листингом =
generate_disasm_line(ea, flg)
print_insn_mnem(ea)
print_operand(ea, n)
get_operand_value(ea, n)
```

```
"mov ebp, 100h"
"mov"
0)"ebp" 1)"100h"
0) 0 1) 256
```

```
GENDSM_*
FORCE_CODE
MULTI_LINE
```

Зависит от арх-ры `get_operand_type(ea, n)`

Произвольное задание инструкции

```
set_manual_insn(ea, insn)
get_manual_insn(ea)
```

n: Номер операнда (с нуля); -1 – все операнды

```
= Взаимодействие с операндами =
toggle_[sign|bnot](ea, n)      # Изменить знак / битовое НЕ
op_[bin|dec|chr|seg|oct|hex|stkvar](ea, n)
op_stroff(ea, n, stId, delta)
op_enum(ea, n, enmid, serial)
op_plain_offset(ea, n, base)
op_offset(ea, n, rtype, tgt, base, tdelta)
```

```
REF_* OFF8 OFF16
OFF32 OFF64 LOW8
LOW16 HIGH8 HIGH16
```

Целевой объект смещения | *Базовый адрес смещения* | *Смещение от целевого объекта*

Произвольный операнд

```
op_man(ea, n, val)
get_forced_operand(ea, n)
```


INFORION
Pavel Rusanov

v.1.0