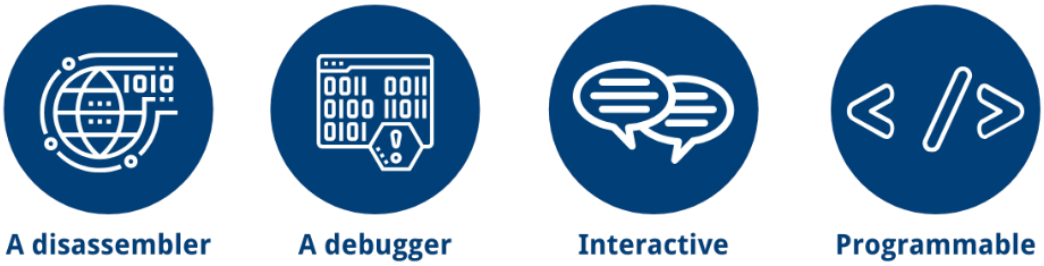
Hex-Rays

IDA PRO – the state-of-the-art binary code analysis tool

IDA Pro is the flagship product of Hex-Rays, the software provider in reverse engineering. Being an interactive and programmable disassembler and debugger, IDA Pro provides excellent quality performance on different platforms and is compatible with many processors. IDA Pro has become the de-facto standard for the analysis of hostile code, vulnerability research and commercial off-the-shelf validation.

IDA Pro comes with different types of licenses: Named, Computer, Floating and Educational license to meet different business' scales and demands of usage.

IDA PRO, in a nutshell



Key features

Multi-processor Disassembler

- Disassembler modules for a large number of processors. The free SDK even allows you to run your custom disassembler;
- Full and extensible interactivity;
- Programmable: IDA can be extended in line with user's own requirement with IDC or IDAPython
- Open plugin architecture: external plugins enable extension of IDA's capability
- FLIRT technology (Fast library identification and recognition technology);
- Code graphing
- Lumina server holds metadata with a large number of well-known functions

Multi-target Debugger

- The debugger adds the dynamic analysis of the information collected statically by the disassembler;
- Offers all the features expected from a debugger and more: "remote" function and tracking. Remote debugger: for Windows, Linux, Mac OS X, and other machines in any combination;

More features and upgrades are introduced along with new IDA version releases!

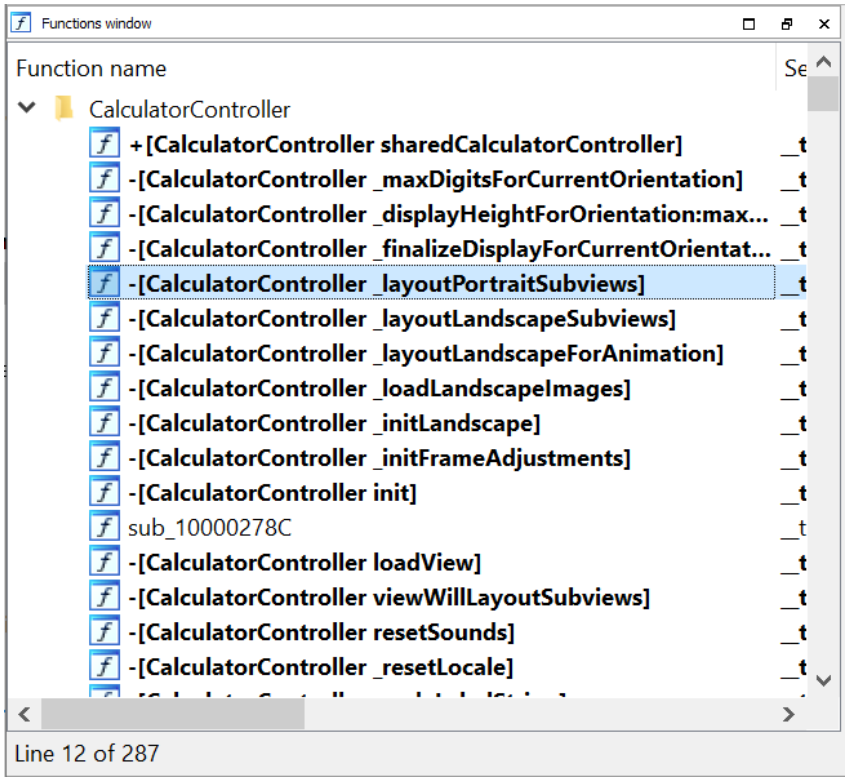
IDA PRO Version 7.5

Release date: May 2020

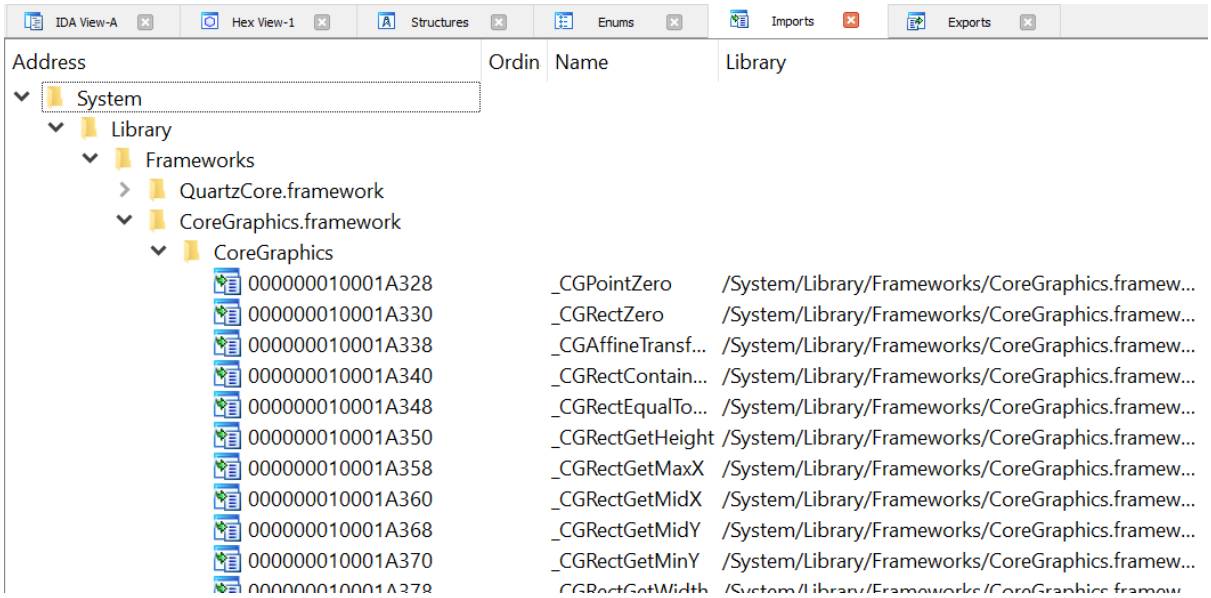
Highlights of new features and improvements:

1. Tree-like folder view:

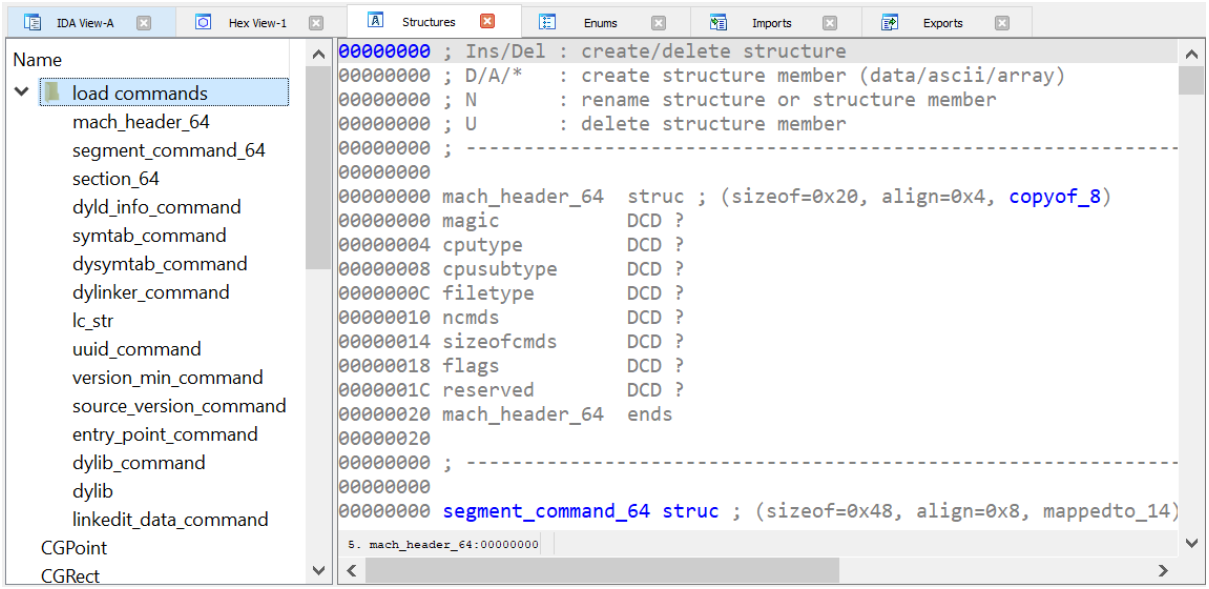
Functions and Names



Imports:



Structures

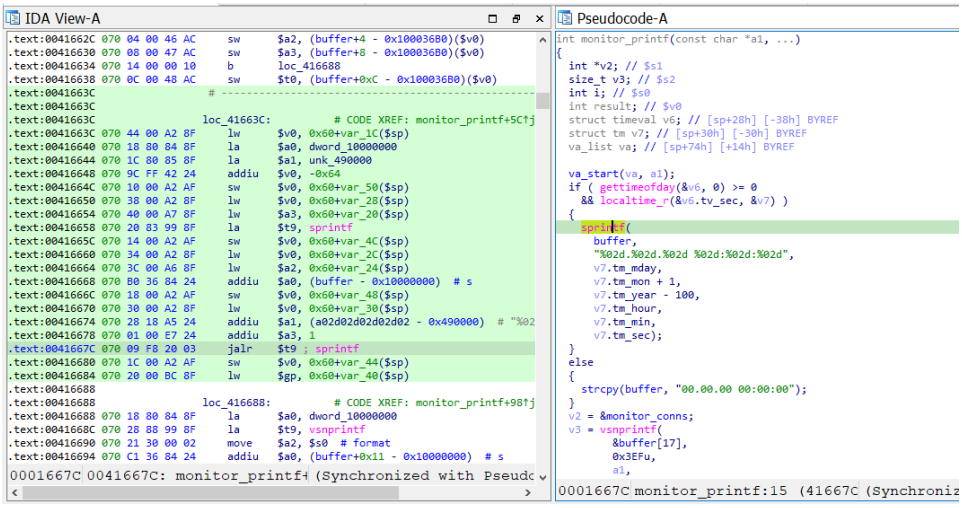


For Structures and Enums, the tree panel is shown by default, for other views it can be enabled via the "Show Folders" context menu item. Users can create, rename and delete folders, and move items between them. This helps organizing information when dealing with large binaries.

2. MIPS decompiler:

A new decompiler has been added to our lineup. Any 32-bit MIPS binary supported by IDA can be decompiled, including compact encodings. The infamous delay slots are handled transparently and seamlessly.

Big-endian MIPS32 code:



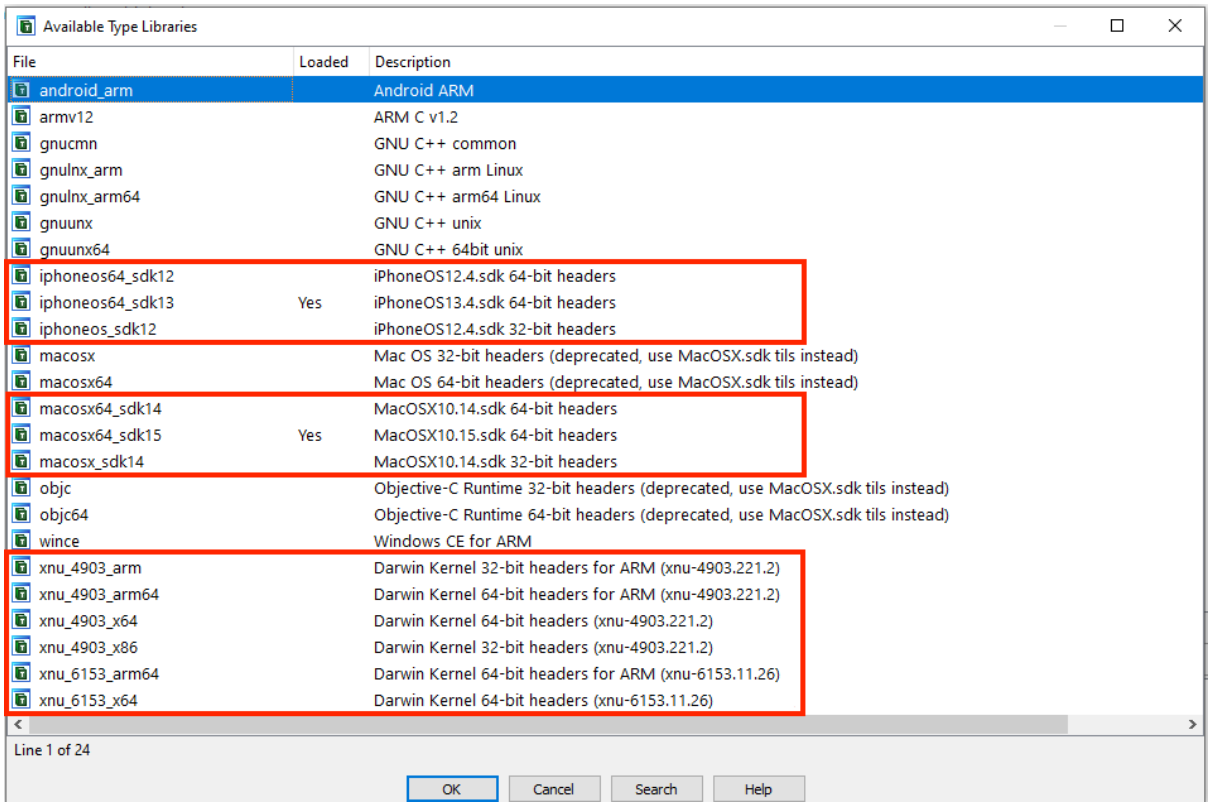
3. Lumina for MIPS and PPC:

Lumina function is now available for MIPS and PPC binaries.

4. iOS/macOS improvements:

Type libraries with the most major APIs and additional frameworks from macOS and iPhone SDKs were added. They are especially useful when paired with Hex-Rays decompiler.

List of initially available type libraries:



FULL UPDATE LIST OF IDA PRO 7.5:

Processor modules:

- ARC: added support for ARCv2 EM instruction set
- ARM: added an option to control detection of 32-bit constants loaded by scattered pairs of MOVW+MOVT instructions
- ARM: improved detection of functions with delayed prolog setup
- MIPS: added support for multi-GOT binaries ($gp can have different values in different parts of the binary)
- V850/RH850: don't create functions for PIC calls (to next address)
- PPC: added many new instructions from e200 cores (NXP MPC57xx, ST SPC58xx):
  - Cache Bypass Storage (lbdcbx lhdcbx lwdcbx stbdcbx sthdcbx stwdcbx dsncb)
  - E200z490 (AIOP) instructions (e_lqw e_stqw e_ldwcb e_ldbrw e_byterevw and more)
  - MPU instructions (mpure, mpuwe, mpusync)
- PC: added support for endbr instruction in prolog analysis
- PC: added decoding of WAITPKG instructions (TPAUSE, UMONITOR, and UMWAIT)
- PC: added decoding of TSX instructions (XRESLDTRK and XSUSLDTRK)
- PC: added decoding of instructions CLDEMOTE, ENCLV, SERIALIZE
- PC: added decoding of Direct Store instructions (MOVDIRI and MOVDIR64B)
- PC: added decoding of MCOMMIT and RDPRU instructions (AMD Zen2)

File Formats:

- AMIGA: implement rebasing for Amiga hunk file loader (contributed by Vladimir Kononovich)
- ELF: ignore internal compiler symbol gcc2_compiled
- ELF: pc: handle PLT stubs in binaries compiled with Intel CET support (-fcf-protection)
- ELF: accept files with PT_LOAD segments running over end of file
- ELF: MIPS: implemented relocations R_MIPS_GOT_PAGE, R_MIPS_GOT_OFST
- ELF: MIPS: add support for MIPS64 complex relocations
- MACHO: allow the user to configure the type libraries loaded for new macho files. see TIL_CONFIG in macho.cfg
- TDS: added support for tds files concatenated with the exe file

Installer:

- Default to Python 3; bundle Python 3.8.2 with Windows installer

Debugger:

- Debugger: added support for Bochs 2.6.10
- Debugger: added debugging support for Zilog Z80 processors
- Debugger: gdb: improve debugging of multi-thread programs
- Debugger: ios: added iPhone SE 2 to list of known devices/li>
- Debugger: PIN: support building pintool with pin 3.13
- Debugger: xnu: improved ktrw support. breakpoints/watchpoints/registers now work as expected with ktrw, using the “Corellium-ARM64” configuration. no other manual setup is needed.

Kernel / Misc.:

- Demangler: add c++20 spaceship and co_await operators for VC++ and GCC
- KERNEL: add std::_Xlength_error() to the list of no-returning functions
- Lumina: Lumina functionality is available for MIPS and PPC binaries

FLIRT / TILS / IDS:

- TIL: introduced new macosx type libraries, built directly from headers in MacOSX.sdk/iPhoneOS.sdk (including all Objective-C and C++ Frameworks). see macosx_sdk*.til/iphoneos_sdk*.til
- TIL: introduced new type libraries specifically for XNU kernel and KEXT binaries, built directly from the XNU source code. see xnu.til/xnu_arm.til
- FLIRT: Added MFC signatures for vc1424 (Visual Studio 2019.4)
- FLIRT: Added MFC signatures for vc1425 (Visual Studio 2019.5)
- FLIRT: ICL: Added signatures for icl200 (Intel C++ 20.0
- FLIRT: ICL: Added signatures for icl201 (Intel C++ 20.1)
- FLIRT: VC: Added signatures for vc1424 (Visual Studio 2019.4)
- FLIRT: VC: Added signatures for vc1425 (Visual Studio 2019.5)

User Interface:

- UI: many IDA views now provide an alternative, tree-like folder view
- UI: added actions to search for register definition or register use (Shift+Alt+Up, Shift+Alt+Down)

- UI: it is now possible to add, delete, enable & disable breakpoints from the 'Function calls' widget
- UI: The "Breakpoints" chooser now also reports the state (Enabled/Disabled/Unresolved) in a column, instead of only through the icon.
- UI: within a session, IDA will by default remember and restore dialogs positions & sizes (configurable through RESTORE_DIALOGS_GEOMETRIES)
- UI: debugger: the current thread is now shown in bold
- UI: debugger: include the hostname and port number in the error message about failed connection
- UI: removed the limitation on syncing similar views (e.g. now it's possible to sync 2 idaviews)
- UI: show filename of the file being loaded during the loading process
- UI: "create struct from data": when used inside a struct, ignore dummy field names like "field_xxx"
- UI: added get_synced_group(), to retrieve information about what widgets are synchronized.

Plugins:

- Pdb: speed up loaing types from big PDBs
- Dscu: introduce a submenu for dyldcache handling (File>Load file>DYLD Shared Cache Utils)
- Dscu: allow branch islands to be loaded from the ui (File>Load file>DYLD Shared Cache Utils>Load branch island)
- Dscu: allow loading one or more modules from a given module's dependency list (File>Load file>DYLD Shared Cache Utils>Load dependency)
- Dscu: allow module headers to be loaded individually from the dyldcache
- Dscu: allow the formatted dyld header to be loaded manually
- Dscu: allow the user to load single sections from any module individually
- Dscu: convert the module chooser to a multi-chooser. now multiple dyldcache modules can be loaded at once (File>Load file>DYLD Shared Cache Utils>Load module)
- Export data: allow user to change the variable name when exporting data as a C array
- Export data: when exporting an item as a C array, use the array variable name as the filename
- Objc: improve decompilation of objc_msgSendSuper() call sites
- Svdimport: new plugin to load and apply ARM CMSIS compliant SVD files with memory register definitions

Decompilers:

- Hexrays: added actions "Remove function argument", "Remove return value" (default hotkey Shift-Del)
- Hexrays: added a variable annotation: BYREF, for the variables whose address is taken
- Hexrays: added action AddRemoveReturn (Ctrl-Shift-R)
- Hexrays: added an option to correctly handle _readflags(); since the results are not really readable, this option is off by default
- Hexrays: added mbl_array_t::save_snapshot() to be used by third-party plugins
- Hexrays: changed the default hotkey of "jump to global xref" to Ctrl-Alt-X. (Ctrl-X was not working in the struct view on macOS)
- Hexrays: arm: support atomic intrinsic instructions from ARMv8.1-A (LDADDAL, CASAL etc.)
- Hexrays: added logic to find enum members in switch cases

- Hexrays: added config option DISABLE_USERCALL to disable automatic generation of usercall prototypes
- Hexrays: improved recognition of CONTAINING_RECORD for structures with one pointer member
- Hexrays: improved recognition of struct member references
- Hexrays: open_pseudocode() now accepts a set of flags for finer control over how to open pseudocode views
- Hexrays: pc: added support for endbr instructions
- Hexrays: ppc: improve handling of soft float compiler helpers
- Hexrays: support some inlined string/memory operations for wide (16-bit) characters
- Hexrays: use standard "Rename address" dialog in pseudocode view to rename global names

Scripts & SDK:

- SDK: extend processor modules, plugins and loader API to be able to use a C++ class for internal implementation
- SDK: added enumerate_files2() to enumerate files using a visitor class
- SDK: added FC_CALL_ENDS flag for qflow_chart_t() to return basic blocks terminated by call instructions
- SDK: added register_cfgopts() which can be used to enable third-party config parameters in process_config_line()
- SDK: added the 'adding_segm' event
- SDK: added the 'func_deleted' event
- SDK: added find_reg_access()
- SDK: qflow_chart_t now computes graph predecessors by default. FC_NOPREDS flag can be used to skip this computation if necessary
- SDK: renamed bitrange_t::combine() -> bitrange_t::apply_mask()
- SDK: exported alloc_kreg/free_kreg functions for decompiler API
- SDK: exported process_config_directive; also renamed process_config_line in idc/python to process_config_directive
- SDK: simplified handling of custom refinfo types; now refinfo_t::type() returns a type with the REFINFO_CUSTOM bit for custom refinfos and refinfo_t::set_type() sets both the type and the REFINFO_CUSTOM bit;
- IDC: added clear_selection()
- IDC: added convenience macros to set the application bitness (inf_set_64bit(), inf_set_32bit())
- Idc: added stristr(), tolower(), toupper()
- IDAPython: added an example showing how to retrieve register information from the context menu
- IDAPython: ida_bitrange is now available

BUGFIXES:

- BUGFIX: "bad event during undo" could occur in some cases
- BUGFIX: "find next error" could crash IDA
- BUGFIX: "ida -I1" was modifying a wrong registry key when trying to set itself as the systemwide just-in-time debugger
- BUGFIX: ARM: A64 LDARP instruction was printed with an incorrectly duplicated operand
- BUGFIX: ARM: IDA could show wrong values if instruction simplification was enabled and instrucions with shifted immediate values were present
- BUGFIX: ARM: The A64 instruction CRC32W was printed with an unnecessary .W suffix

- BUGFIX: compile_idc_snippet() could fail if the snippet was ending with a comment and no newline
- BUGFIX: cursor position in the list of xrefs to stkvars was not preserved
- BUGFIX: debugger: a malicious client could invoke commands on a password-protected debug server without a password
- BUGFIX: debugger: IDA could crash with interr 40052 when exiting while process is suspended with tracing enabled
- BUGFIX: debugger: IDA could exit with internal error 40038 if erasing a breakpoint from the process failed unexpectedly
- BUGFIX: debugger: IDA could fail to attach through GDB to a running instance of QEMU
- BUGFIX: debugger: IDA could INTERR with 64-bit GDB flags register
- BUGFIX: debugger: in rare cases IDA could crash when using Appcall in win32 debugger
- BUGFIX: debugger: ios debugger could fail to handle read/write breakpoints in multithreaded situations.
- BUGFIX: debugger: linux: the base of segment registers was calculated incorrectly in x86_64
- BUGFIX: debugger: PPC: when debugging VLE code, IDA could put breakpoints at wrong locations
- BUGFIX: debugger: values of Dn registers on ARM32 platform would not be available
- BUGFIX: debugger: when attaching to some Windows 10 systems using Windbg backend, IDA would appear to hang
- BUGFIX: debugger: win32: On Windows 7, IDA could incorrectly rebase the database if the executable was mapped into the address space a second time (can happen e.g. when displaying the icon in a File Open dialog)
- BUGFIX: decompiler: assigning to a part of a variable could be erroneously translated as assigning to the whole variable
- BUGFIX: decompiler: changed the hotkey for "global xrefs" to Ctrl-X because Shift-X does not work well in all contexts (for example, in choosers)
- BUGFIX: decompiler: decompiler could lose instructions which modified its operands
- BUGFIX: decompiler: fixed a crash on decompilation failure when COLLAPSE_LVARS=YES in hexrays.cfg
- BUGFIX: decompiler: fixed interr 52329, which could occur if a enum type was renamed after its application in the decompiler
- BUGFIX: decompiler: fixed numerous internal errors
- BUGFIX: decompiler: IDA could crash with unhandled exception on opening a database which was saved after using the decompiler
- BUGFIX: decompiler: in some cases "Cancel" button did not stop the decompilation
- BUGFIX: decompiler: interr could occur if a parenthesis was used in a variable name
- BUGFIX: decompiler: it could be required to press 'Escape' twice in order to cancel a decompilation requested by jumping to an address
- BUGFIX: decompiler: it was impossible to input the negative number for the shifted value in the "convert to struct*" dialog
- BUGFIX: decompiler: ppc instruction mulhd was decompiled incorrectly
- BUGFIX: decompiler: pressing enter at the end of the very first line of the function body would not add an empty line as it should
- BUGFIX: decompiler: renaming the same variable twice from two different pseudocode windows could cause an erroneous warning
- BUGFIX: decompiler: some forced variables were not applied correctly
- BUGFIX: decompiler: some lvar mappings would be ignored by the decompiler
- BUGFIX: decompiler: some SSE2 instructions were decompiled to wrong intrinsics
- BUGFIX: decompiler: when canceling a jump from "Pseudocode-A" to a new function, canceling decompilation could cause IDA to switch to "IDA View-A"

- BUGFIX: demangler: for old borland mode (v < 5.5) some types in template arguments were demangled incorrectly
- BUGFIX: DWARF: The DWARF plugin could complain about invalid data for some Golang binaries
- BUGFIX: DWARF: The DWARF plugin could enter an inconsistent state and bail out upon certain constructs
- BUGFIX: DWARF: The DWARF plugin could fail to parse certain constructs involving similarly-named typedefs, to various templates instantiations
- BUGFIX: DWARF: The plugin could create the same parameter multiple times, if certain (GCC) constructs were used to specify their const value
- BUGFIX: ELF: MIPS: improve handling of the special symbol "_gp_disp"
- BUGFIX: ELF: PLT stubs could be truncated and marked as no-return in some MIPS files, resulting in bad analysis
- BUGFIX: ELF: some ARM shared objects could fail to resolve external symbols (imports)
- BUGFIX: enum radix was not immediately propagated from the enum view to the local types
- BUGFIX: fixed a random interr 30143 that was occurring when attaching to a WoW64 application that was generating lots of exceptions
- BUGFIX: fixed erroneous internal error 1544 that could occur after a debugger session
- BUGFIX: gdb debuggers could interr 30044 in multithreaded situations.
- BUGFIX: GDB would not mask exceptions even if configured to do so
- BUGFIX: GDB would not respect the user's request when manually resuming after exceptions
- BUGFIX: GDB: LR was incorrectly set as instruction pointer for PPC configurations (correct register is PC)
- BUGFIX: hexview: editing undefined byte and setting its value to 0xFF, could fail to show the value properly
- BUGFIX: IDA analysis could loop indefinitely when analyzing some switch patterns produced by clang (e.g. in chrome.dll)
- BUGFIX: IDA could crash in case of a network error or if a remote GDB target did not support/report threads
- BUGFIX: IDA could crash on exit when cleaning the leaked type objects (e.g. after a decompiler error)
- BUGFIX: IDA could crash when debugger flag names were used as variables in IDC scripts
- BUGFIX: IDA could crash when loading a new database with autoanalysis in progress
- BUGFIX: IDA could crash when using watches during debugging
- BUGFIX: IDA could fail to restore some segment register areas
- BUGFIX: IDA could INTERR(40662) with C++ plugins that provide a PCF_EA_CAPABLE place_t implementation
- BUGFIX: IDA could produce a fatal error when applying a function prototype with __spoils list which included ARM64 Xnn registers
- BUGFIX: IDA would exit without any error message if a wrong -r switch was provided in the command line (for example, if the remote server was not reachable)
- BUGFIX: idapyswitch on Windows could not distingush separate Python installs with the same version
- BUGFIX: idapyswitch would not handle Python versions installed by macports
- BUGFIX: IDAPython: after showing forms (or simply calling 'set_script_timeout()'), it could happen that the "Running Python script" wait dialog wouldn't show anymore for long operations
- BUGFIX: IDAPython: calling add_segm_ex with a None segment, could crash IDA
- BUGFIX: IDAPython: func_t.referers array was not usable from Python
- BUGFIX: IDAPython: ida_dbg.get_current_source_file() was not usable

- BUGFIX: IDAPython: ida_dbg.get_process_options() was not usable
- BUGFIX: IDAPython: ida_funcs.func_t.points was unusable (and could cause IDA to crash)
- BUGFIX: IDAPython: ida_funcs.func_t.regargs was not usable
- BUGFIX: IDAPython: ida_idp.IDP_Hooks::ev_set_idp_options (and thus ida_idp.processor_t::ev_set_idp_options) was unusable
- BUGFIX: IDAPython: ida_kernwin.Form instances could raise exceptions when using GetFieldValue on certain non-input fields
- BUGFIX: IDAPython: ida_struct.struc_t.get_member() could return pointer to invalid data
- BUGFIX: IDAPython: ida_struct.struc_t.members was not usable as it only ever allowed accessing the first member
- BUGFIX: IDAPython: idapyswitch on linux could fail to be used again after being used to set target library to 'libpython3.so'
- BUGFIX: IDAPython: idapyswitch would fail to link on Windows when using public source tree with the IDA SDK
- BUGFIX: IDAPython: idc.get_inf_attr() could raise an exception due to improper type comparison with scripts showing a wait dialog
- BUGFIX: IDAPython: idc.GetLocalType() could report a UnicodeDecodeError
- BUGFIX: IDAPython: idc.py: "is not "" is not valid in Python 3.8.1
- BUGFIX: IDAPython: in some circumstances, building a GraphViewer could cause a very cryptic "AttributeError: 'Graph' object has no attribute 'id'" error
- BUGFIX: IDAPython: insn_t.auxpref was limited to 16 bits, instead of correct 32
- BUGFIX: IDAPython: issuing a 'ida_search.find_binary' call while debugging and if ida_kernwin.UI_Hooks were hooked, could cause IDA to hang
- BUGFIX: IDAPython: performing an ida_idd.Appcall on a function that takes an 'int *', and in order to do so using a construct of Appcall.int64() + Appcall.byref() to construct the argument, could yield incorrect results
- BUGFIX: IDAPython: processor modules, loaders & plugins should have their '__file__' properly set, since they are not using the '__main__' namespace
- BUGFIX: idc: it was impossible to call a function through a pointer stored in a class member: obj.funcptr = func; obj.funcptr()
- BUGFIX: installer: idapyswitch would incorrectly ignore valid Python installs as "unusable AppStore Python" on Windows 7
- BUGFIX: M16C: addresses were not truncated to 32 bits when using IDA64
- BUGFIX: M740: bra and jmp must stop the execution flow
- BUGFIX: MACHO: load commands with ids larger than LC_DYLD_ENVIRONMENT were formatted incorrectly in the header segment
- BUGFIX: mips: fixed decoding of the 'break' insn;
- BUGFIX: mips: fixed decoding of the 'trunc.w/l' for microMIPS;
- BUGFIX: mips: fixed endless loop if a call delay slot was changing $t9;
- BUGFIX: mips: fixed the setting of initial $gp value
- BUGFIX: mips: implemented support for get_reg_accesses
- BUGFIX: MIPS: microMIPS 16-bit lw/st instructions were decoded incorrectly (with signed offset instead of unsigned)
- BUGFIX: Objective-C step-into action could fail on MacOSX10.15/iOS13.
- BUGFIX: On Windows, IDA could crash on some IDBs if the current codepage was changed to 65001
- BUGFIX: PC: IDA would appear to hang if a very long sequence of nops was present in the middle of a function
- BUGFIX: PDB: in some cases the types loaded from PDB file ("Types only") would be wrong and may cause interr
- BUGFIX: PDB: the size of enum was set incorrectly

- BUGFIX: PE: files with IAT lying outside of .idata could result in empty Imports list (even though actual import pointers were properly renamed)
- BUGFIX: PE: when loading a mixed .NET file as native PE, imports list would be empty when using default options
- BUGFIX: PIN: in some cases IDA did not refresh memoty layout
- BUGFIX: SDK: during debugging, opening the context menu on the register label wouldn't provide the register name to the action_update_ctx_t, as it would on the register value
- BUGFIX: SDK: http_get() was buggy and not reporting a failure if the connection was not established
- BUGFIX: the 16-bit counter that was used for the number of function tail parents could overflow for some huge idbs
- BUGFIX: the iOS debugger could fail to handle a watchpoint after it was hit frequently (100+ times in the same session).
- BUGFIX: ui/qt: Canceling editing of a type in the "Local types" view, could cause it to be reverted to a different state than it was before
- BUGFIX: ui/qt: double-clicking in the "Output window", could fail to jump in the right place, if a very large number of lines was present in the output
- BUGFIX: ui/qt: set_viewer_graph() was not working
- BUGFIX: ui/qt: Some messages in the "Output window" could be truncated in case very long scripts were run
- BUGFIX: ui/qt: when holding the left mouse button down, scrolling with the mouse wheel would clear the selection (if it existed.)
- BUGFIX: ui: 'make array' was not preserving the operand representation
- BUGFIX: ui: A synced pseudocode view could in certain situations fail to show up-to-date contents
- BUGFIX: ui: calling 'unregister_action' for some core IDA actions, could cause IDA to crash
- BUGFIX: ui: current function was not always reanalyzed after manually editing a stack change point which could result in unbalanced stack
- BUGFIX: ui: IDA would crash if "attach to process" dialog was cancelled when working without a database
- BUGFIX: ui: In "Hex View-1", partially editing a byte, then calling "Undo", and then entering edit mode again (by pressing F2), woild cause the partial edit to show again
- BUGFIX: ui: list of patched bytes would be empty when patching a rebased program (e.g. during or after debugging)
- BUGFIX: ui: Rejecting the "String window"'s "Setup" dialog would cause the list of strings to be recomputed anyway
- BUGFIX: ui: the forms change callback was not called for color button changes
- BUGFIX: UI: using "quadro word" in context menu would create a float
- BUGFIX: ui: when in the "Enums" view, pressing <Enter> with cursor on an 'XREF: <function name>" wouldn't jump
- BUGFIX: ui: when re-creating a chooser with a different number of columns, it could happen that some columns were invisible
- BUGFIX: ui: when starting with '-A' (i.e., batch mode), IDA would only show the "Output window" on the desktop
- BUGFIX: Under certain (very rare) circumstances, IDA could freeze while calculating a hint
- BUGFIX: undo: fixed a bug when undoing the debugger segment, added the recording of dbgmem_config
- BUGFIX: windbg: ordinary breakpoints located in the same memory page as page breakpoints would be handled incorrectly

Hex-Rays

IDA PRO Version 7.5 – Service Pack 1

Release date: 19th June 2020

Highlights: This Service Pack was released to improve user experience especially for newly released features such as the tree-like folder view function and the MIPS decompiler.

Full changelist: https://www.hex-rays.com/products/ida/news/7_5sp1/

IDA PRO Version 7.5 – Service Pack 2

Release date: 28th July 2020

Highlights: This release fixes some immediate issues with the new macOS11/iOS14 binaries and focuses principally on enhancing the static analysis for new file formats. MH_FILESET kernelcache format is fully supported, Objective-C metadata is improved and type libraries for MacOSX11.0.sdk and iPhoneOS14.0.sdk was added.

Full changelist: https://www.hex-rays.com/products/ida/news/7_5sp2/

IDA PRO Version 7.5 – Service Pack 3

Release date: 28th October 2020

Highlights: This service pack introduces a handful of new and interesting features specific to the soon-to-be-released macOS 11 (Big Sur) and provides fixes for numerous minor issues. macOS11 kernel debugging with VMware Fusion 12 and symbolication of MH_FILESET kernelcaches were both improved.

Full changelist: https://www.hex-rays.com/products/ida/news/7_5sp3/