IDA 7.6 What's new?

Full list of changes and new features:

NB: some items may have been already mentioned in IDA 7.5 SP1-3 release notes

Processor modules:

- 6502: disassemble 65C02-specific opcodes STP and WAI
- 68K: display ranges of Dx and Ax registers explicitly in the movem instruction
- ARM: add support for ARMv8.4-RCPC instructions (LDAPUR, STLUR)
- ARM: add support for golang ABI
- ARM: added support for parsing ARM64 exception handler information in PE files
- ARM: condition flags (ZF, CF etc.) can be used as argument locations
- ARM: decode ARMv8.5-A BTI instruction
- ARM: decode ARMv8.5 barrier instructions
- ARM: fixed switch with CSEL
- ARM: improve detection of functions in binaries using PAC or BTI
- ARM: improve propagation of pointer types and names loaded using LDR(literal) instructions
- ARM: improve switch detection in Thumb code produced by ARM 5.x MDK compiler
- ARM: support ARMv8.5-A Memory Tagging Extension (MTE) instructions
- ARM: support half-precision floating (FP16) operands
- AVR: support AVR XMEGA family chips which do not have general-purpose registers in data memory (contributed by Lukas Kuzmiak from insighti)
- AVR: print operands for instructions movw, adiw, sbiw as register pairs
- MIPS: add support for EABI32
- MIPS: support EVA load/store instructions (LBE, LBUE, LHE, LHUE, LWE, SBE, SHE, SWE, CACHEE, PREFE, LLE, SCE, LWLE, LWRE, SWLE, SWRE).
- MIPS: support interAptiv custom instructions (MIPS16e2 COPYW/UCOPYW, MIPS32 SAVE/RESTORE)
- MIPS: support the MIPS16e2 Application-Specific Extension instructions (used in the InterAptiv core)
- PC: add FXSAVE64 and FXRSTOR64 instructions
- PC: add support for AVX512 VAES (EVEX-encoded AES-NI instructions)
- PC: add support for stack-based golang abi
- PC: add VGF2P8AFFINEINVQB, VGF2P8AFFINEQB and VGF2P8MULB instructions
- PC: decode sysenter and sysexit instructions in long mode (they're supported on Intel processors)
- PC: recognize and handle calls to retpoline thunks (__x86_indirect_thunk_rXX)
- PPC: add support for EFP(Embedded Floating Point) 2.0 instructions efdmin, efdmax, efdsqrt, efdcfh, efdcth
- PPC: improved recognition of position-independent switches
- PPC: improved recognition of switches that use clrlwi to mask off the number of cases
- RISCV: new processor module (initially supporting RV32GC and RV64GC)
- RL78: new processor module

File Formats:

- DWARF: fix detection of golang files, improve use of golang-specfic DWARF info
- DWARF: support batch-loading dwarf files from a macOS11 KDK into an MH_FILESET kernelcache idb

- ELF: mips: handle the most frequently used microMIPS relocations
- ELF: mips: process MIPS16 relocations
- ELF: .gnu_debugdata (MiniDebugInfo) sections are now recognized and parsed
- MACHO: handle dyld slide info v4 (used in WatchOS dyld_shared_cache_arm64_32)
- MACHO: handle LC_DYLD_EXPORTS_TRIE in macOS11/iOS14 binaries
- MACHO: ignore the dummy symbol radr://5614542 that may be added by the strip(1) tool
- MACHO: improve analysis of dyldcache files from macOS11/iOS14
- MACHO: improve handling of threaded pointers in iOS kernelcaches
- MACHO: introduce type libraries for MacOSX11.0.sdk and iPhoneOS14.0.sdk
- MACHO: native support for compressed kernelcaches (lzfse, lzss, lzvn), optionally wrapped in img4 container
- MACHO: parse LC_DYLD_CHAINED_FIXUPS for arm64e binaries. this should heavily improve the analysis for iOS 14 binaries compiled with Xcode 12
- MACHO: support new MH_FILESET kernelcache format from macOS 11
- MACHO: support symbolication of macOS11 kernelcaches that link against the boot/sys kext collection
- PE: rename standard dummy control flow guard thunks (_guard_check_icall_nop, _guard_dispatch_icall_nop)

Debugger:

- bochs: updated ida to support bochs 2.6.11
- dalvik: added device connection string (ADB_CONNECT), to simplify Corellium usage
- dalvik: added option JDWP_WAIT_FOREVER to wait for JDWP reply infinitely, this helps to debug the shared library of Dalvik application
- debugger: don’t exit IDA if creating a debugger segment fails
- debugger: win32: add ‘Stack’ column and ‘Follow in stack view’ context menu to the “SEH list” window
- debugger: macOS: add servers for debugging arm64 and arm64e applications on Apple Silicon

Kernel / Misc.:

- improved handling of corrupted IDBs: detect when a function object is missing and try to fix it
- kernel: support variable-sized structures in unions
- types: allow specification of stack offsets of function arguments (with __usercall); also, return location can be on the stack
- types: if applying a function type to data which can be a pointer (e.g. dword or qword) then convert the type to a function pointer

FLIRT / TILS / IDS:

- TIL: add _PEB_LDR_DATA to the standard “undocumented” structures for mssdk_* tils
- TIL: added type libraries for xnu-7195

User Interface:

- UI: added a database cache for the strings list
- UI: added NAVBAND_FORCE_GAPS option (defaults to NO), to control whether the navigation band should display gaps between adjacent segments
- UI: added the “Clear…” action to the string list window
- UI: automatically sync new enum types to the idb
- UI: bookmark list can be opened as separate non-modal view which supports folders
- UI: bookmarked lines are highlighted with their own customizable overlay color

- UI: debugger hints would override decompiler hints
- UI: in folders, add cut & paste as an alternative means of moving items around
- UI: it is now possible to expand/collapse multiple folders at once, using Ctrl+Numpad+/Ctrl+Numpad-
- UI: shortened names of many additional views
- UI: save the xrefs window position in the text UI
- UI: the navigation band now highlights the segment the mouse is currently hovering
- UI: the quick filter (in the tabular, or folders views) can be targeted to only 1 column, or all of them (the default)
- UI: The "Create Struct From Data" command will now use demangled method names for the new structure members if used on a vtable (list of function pointers)
- UI: themes: implement @ifdef/@ifndef/@else/@endif preprocessing directives for our .css files
- UI: Using T(structure offset) on a struct member and picking another struct, sets its type to "struct *"

Plugins:

- golang: new plugin to parse and use metadata produced by the Go compiler and Go-specific idioms (pclntab, fixed-length string literals)
- eh_parse: improve EH metadata parsing for x64 and ARMv7/ARM64 PE files
- OBJC: fixed objc:RunUntilMessageReceived action for macOS11/iOS14
- OBJC: improve Objective-C metadata parsing for macOS11/iOS14 (specifically __objc_methlist structures)
- OBJC: plugin will now put all decoded Objective-C types in an "objc" folder in the Local Types view
- PDB: Network communications while looking for PDB file can be restricted (option PDB_NETWORK in pdb.cfg)
- PDB: support new UDT tags generated by VS2019
- swift: enabled swift plugin for elf files too, since they can be encountered in the wild
- TDS: improved handling of long names

Decompilers:

- decompiler: "set call type" is available in more situations
- decompiler: add an intrinsic function for the x87 fbstp instruction
- decompiler: added a rule 'and (x|y), #n => or (x&#n), (y&#n)'
- decompiler: added action "Quick rename"
- decompiler: added an option to decompile library functions too
- decompiler: added convenience functions: locate_lvar(), rename_lvar()
- decompiler: added create_cfunc()
- decompiler: added GENERATE_EMPTY_LINES; it improves readability by introducing empty lines between blocks of code
- decompiler: added global xrefs to entire structs/unions
- decompiler: added hints to 'goto LABEL' instruction
- decompiler: added hxe_build_callinfo: plugins now can provide call prototypes dynamically
- decompiler: added mba_t::set_maturity, it can be used to skip some decompilation passes (to be used with caution)
- decompiler: added new function mba_t::build_helper_call
- decompiler: added new methods to codegen_t: store_operand(), load_effective_address()
- decompiler: added option AUTO_UNHIDE – automatically unhide collapsed items in the pseudocode view when jumping to them
- decompiler: added option HO_NON_FATAL_INTERR: permit decompilation after an internal error
- decompiler: added the distribution rule for or/and

- decompiler: do not use ‘l’ and ‘ll’ and the loop index names anymore
- decompiler: extended ‘split expression’ to support 64-bit ‘not equal’ operator
- decompiler: extended the functionality of global xrefs to all local types, not only to udts and enums
- decompiler: improve decompilation of ternary operator expressions
- decompiler: improve recognition of array stack variables
- decompiler: improved detection of bool return value
- decompiler: improved detection of optimized array references like &buf[i-1]
- decompiler: improved handling of imagebase relative addressing
- decompiler: improved management of guessed types
- decompiler: improved recognition of some signed divisions
- decompiler: improved recognition of x64 strcmp
- decompiler: improved use of return statements, now we create more of them
- decompiler: mips: add support for EABI32
- decompiler: never automap variables having a potentially undefined value
- decompiler: now the decompiler replaces unused call arguments with UNUSED_ARG(); the unused arguments can be marked with the “__unused” keyword
- decompiler: optimize bit test operations on variables with a known set of values
- decompiler: pc: decompile bit operations (bt, bts, btr) on 16-bit operands as custom intrinsics (e.g. _bittest16).
- decompiler: print too long function prototypes on multiple lines
- decompiler: rename local variables more aggressively, to get more meaningful names
- decompiler: slightly improved goto elimination
- decompiler: user-defined cross-references can be used to inform the decompiler about the target of an indirect branch
  Scripts & SDK:

- idapyswitch: add Anaconda 2020.02/11 to the ignore list
- IDAPython: added an example showing how to color a chooser according to its contents
- IDAPython: added an example showing how to toggle pseudocode lines’ background using UI hooks
- IDAPython: added an example showing how to use ida_kernwin.create_menu() to create toplevel menus, or submenus
- IDAPython: added an example showing how to use custom_viewer_jump()
- IDAPython: added an example of using ida_bytes.bin_search()
- IDAPython: exposed ida_ieee module, to deal with internal representation of floating values
- IDAPython: ida_gdl.gdl_graph_t (& subclasses) are now available
- IDAPython: prepare for Python 3.10
- SDK: added DP_SZHINT (and WOPN_DP_SZHINT) for set_dock_pos() to let plugin writers specify a desired size for plugin widgets
- SDK: added set_vftable_ea() to tie a vftable type and the address of its instance in the idb
- SDK: file2base() and mem2base() change storage type to non-sparse
- SDK: IDA now sends a ‘ui_desktop_applied’ event after a desktop has been applied
- SDK: improved create_align() to accept length=0; it will be inferred from the alignment exponent in this case
- SDK: introduced fpvalue_t that represents floating point values in internal format. it replaces ushort[6], which means that the source code of some plugins may require changes. however, binary compatibility is maintained
- SDK: introduced processor_t::ev_update_call_stack
- SDK: renamed confusing inf_is_32bit() -> inf_is_32bit_or_higher(); added inf_is_32bit_exactly(), inf_is_16bit(), inf_get_app_bitness()

Bugfixes:

```
BUGFIX: "push esp/pop reg" was decompiled incorrectly
```

BUGFIX: .NET: some floating point values could be printed truncated

BUGFIX: 68K: extended-precision floating point constants were not displayed correctly

BUGFIX: 78K0S: opcode D5 was incorrectly decoded as INC (should be DEC)

BUGFIX: Alt+T/Ctrl+T searches in tabular/tree views, wouldn't wrap around as they should

BUGFIX: ARM: IDA could destroy a user-defined switch statement

BUGFIX: avr: relative jumps/calls could be truncated on parts with more than 64K of program memory.

BUGFIX: backward searching for bytes could fail in certain cases

BUGFIX: chooser: the ui_get_chooser_item_attrs event was called with the wrong CHOOSER argument

BUGFIX: clear_cached_cfuncs() was not clearing the global xref cache

BUGFIX: Command+M would not minimize the IDA window on macOS, per convention.

BUGFIX: debugger: linux: the collected thread may die prematurely before attaching, ignore it

BUGFIX: debugger: xnu debugger would fail to demangle c++ names after attaching with an empty database

BUGFIX: decompiler SDK: wrong return code in udcall_t::compare()

BUGFIX: decompiler would try to use shortcut "Ins" instead of "I" for the "Edit block comment" action on macOS.

BUGFIX: deleting local types could generate interr 81

BUGFIX: dscu plugin was broken if the user changed the "Input file" field in the Process Options dialog.

BUGFIX: dscu: plugin would work incorrectly after rebasing a dyldcache database

BUGFIX: DWARF: IDA could try to use too much memory on corrupted files before dying with out-of-memory

BUGFIX: DWARF: plugin could cause IDA to crash (stack exhaustion) with some specially crafted input files

BUGFIX: DWARF: plugin could crash IDA (null pointer dereference) with some specially-crafted files

BUGFIX: DWARF: plugin could INTERR and cause IDA to exit on specially crafted files with bad function names

BUGFIX: DWARF: plugin could loop (seemingly) endlessly when encountering a DW_TAG_namespace with a (broken) name whose first character is '#'

BUGFIX: DWARF: plugin could perform a use-after-free on some specially crafted files

BUGFIX: DWARF: plugin could perform a use-after-free during stack unwinding, on some DWARF input files

BUGFIX: DWARF: validate size of compressed sections before trying to load them

BUGFIX: ELF: fixed processing of the R_X86_64_32S reloc;

BUGFIX: ELF: PPC: IDA could crash when loading a corrupted elf file

BUGFIX: find_plugin() would load previously unloaded plugin even with load_if_needed=false

BUGFIX: fixed interr 50729 that could occur after mapping a local variable

BUGFIX: forcing plugin to be unloaded by setting PLUGIN_UNL in the run() method did not work for PLUGIN_MULTI plugins

BUGFIX: IDA could crash at restart time (e.g. when restoring a snapshot) if some of plugins installed post-event visitors

BUGFIX: IDA could crash with an internal error when stepping into functions with long names if "Reconstruct the stack" was enabled in debugger options

BUGFIX: IDA could crash in some cases when using accessibility features (e.g. a screen reader)

BUGFIX: IDA could endlessly loop on some corrupted idbs

BUGFIX: IDA could erroneously complain about "CRC32 mistmatch" when opening legacy IDBs (from IDA 5.x or earlier)

BUGFIX: IDA could fail with internal error 20078 on corrupted ELF files

BUGFIX: IDA could produce an internal error while analyzing database after rebasing

BUGFIX: IDA would crash when loading an ARM64 driver if the default debugger was set to windbg

BUGFIX: IDA would create many useless *_hidden segments when loading an XNU kernelcache.

BUGFIX: IDA would not properly keep track of the imagebase for dyldcache idbs.

BUGFIX: IDA would try to allocate huge amount of memory when loading a corrupted elf file

BUGFIX: idapyswitch could fail to detect Python3 versions installed via homebrew on macOS

BUGFIX: idapyswitch would not respect the "-r" switch (dry run)

BUGFIX: IDAPython: 'coding: ' comments were not respected when loading a script file.

BUGFIX: IDAPython: added a 'bytes' property to Python classes wrapping C++ arrays

BUGFIX: IDAPython: IDA could exit silently on startup if the Python runtime called exit() during intialization (can happen with some Python distributions like Anaconda). Now we try to detect such situation and show an explicit error message.

BUGFIX: IDAPython: ida_bytes.bin_search documentation was lacking

BUGFIX: IDAPython: ida_bytes.next_visea, ida_bytes.prev_visea were not available

BUGFIX: IDAPython: ida_hexrays.mop_t.[_make_cases|_make_callinfo|_make_pair|_make_insn] could crash IDA

BUGFIX: IDAPython: ida_hexrays.mop_t.make_fpnum was unusable

BUGFIX: IDAPython: ida_ida.AF_FINAL had value -0x80000000 instead of 0x80000000

BUGFIX: IDAPython: ida_kernwin.del_hotkey could delete the wrong action, and cause IDA to crash

BUGFIX: IDAPython: ida_name.MNG_* and ida_name.MT_* values were not exposed

BUGFIX: IDAPython: ida_search.SEARCH_UNICODE was not available after IDA 7.0, while ida_search.find_binary() still is

BUGFIX: IDAPython: idapyswitch now supports Python 3.9

BUGFIX: IDAPython: IDAPython-on-Python3.9 was unusable because the result of evaluating expressions was not printed

BUGFIX: IDAPython: idautils.Strings.setup() would not apply the 'ignore_instructions' parameter

BUGFIX: IDAPython: if a 'nav colorizer' would return a long that couldn't be converted into 32-bits, IDA would fail reporting the issue in a timely manner, leaving it for later Python code to fail

BUGFIX: IDAPython: if a Python loader & a Python processor module had the same file name, the processor module couldn't be loaded

BUGFIX: IDAPython: internal error 30615 could happen if Python intialization failed

BUGFIX: IDAPython: using ida_kernwin.choose_find() with a non-IDAPython chooser, would crash IDA

BUGFIX: IDAPython: using ida_kernwin.set_nav_colorizer() could cause IDA to crash at exit-time

BUGFIX: IDAPython: when using Python 2, scripts with magic 'encoding' comment, could fail to run

BUGFIX: in ev_renamed event, the 'local_name' could be wrongly reported as 'true' if a local label was requested to be deleted but ida automatically replaced it with a dummy name. this may happen if the item with the name had xrefs to it

BUGFIX: it was impossible to pass REFINFO_SUBTRACT and REFINFO_SIGNED into op_offset();

BUGFIX: kernel: functions could be restored incorrectly from a corrupted IDB

BUGFIX: loading single modules from a dyldcache was unusually slow on OSX Catalina.

BUGFIX: M68K: some fmovem variations were disassembled incorrectly

BUGFIX: mac debugger could fail to attach to a process after previously detaching from it.

BUGFIX: mac debugger could fail to load symbols from system dylibs (revealed by macOS11 beta 4).

BUGFIX: mac debugger would show "Input file is missing" error when debugging a dyldcache lib on macOS11.

BUGFIX: mac/ios/xnu debuggers would create tons of meaningless debugger segments.

BUGFIX: macho loader could fail to load a correct SDK til in some cases.

BUGFIX: MIPS: 'search for register access' could hang

BUGFIX: MIPS: IDA could show incorrect names and comments for CP0 registers when select was not zero

BUGFIX: modifying an attribute of a function argument (e.g. adding __hidden) would be saved in the database but would not be immediately reflected in the disassembly

BUGFIX: objc analysis could fail due to arm64e tagged pointers.

BUGFIX: objc plugin could create invalid xrefs to Objective-C methods during decompilation (IDA-2487)

BUGFIX: objc plugin could fail to create structures in the database after a rebase operation.

BUGFIX: on windows idat would let the operating system to handle some Ctrl- keys, rendering them unusable in ida

BUGFIX: PC: 'in' instruction in 64-bit mode uses EAX and not RAX register.

BUGFIX: PC: endbr64 instruction is suported in 32-bit and 16-bit modes

BUGFIX: PC: extra prefix was not always displayed on a separate line.

BUGFIX: PC: fix decoding of instructions that use VEX.W/EVEX.W in 32-bit mode

BUGFIX: PC: fix operand type for long mode 'call far' to dt_tbyte (2-byte selector plus 8-byte offset)

BUGFIX: PC: fix operand types for many VEX-encoded AVX/AVX2 instructions

BUGFIX: PC: fixed decoding of some AVX instructions in 32-bit mode

BUGFIX: PC: huge functions could cause a simplex algorthm failure

BUGFIX: PC: in 32-bit mode, the target must be truncated to 16 bits if the instruction uses prefix 66 and/or 67.

BUGFIX: PC: in 64-bit mode the operand size for near call is forced to 64-bits.

BUGFIX: PC: in MOVSXD r16, r/m16 instruction, the first operand is a 16-bit register.

BUGFIX: PC: parse_reg_name() could return the wrong result for XMM/YMM/ZMM registers

BUGFIX: PC: processor specific options were not undone upon Ctrl-Z

BUGFIX: PC: some FMA instructions were not decoded in 32-bit mode

BUGFIX: pdb: COFF: subsection SYMBOLS of ".debug$T" may have zero size, use the remaining bytes of section ".debug$T"

BUGFIX: pdb: do not interr on bad-formed udt

BUGFIX: pdb: fix looping in LF_MODIFIER leaf

BUGFIX: pdb: fixed out-of-bounds read array access

BUGFIX: pdb: fortify TPI/IPI streams header parsing

BUGFIX: pdb: in rare cases the last bytes of a stream could be read incorrectly

BUGFIX: pdb: IPI stream could be parsed incorrectly

BUGFIX: pdb: size of a scalar type could be wrong

BUGFIX: pdb: SKIP symbol could be harmful in specially-crafted pdb-file

BUGFIX: PPC: e_ori. with the condition record bit was wrongly simplified to e_nop.

BUGFIX: rebasing a dyldcache idb could break the analysis because relocations were not applied to pointers in the slide info

BUGFIX: rebasing the program by an odd number of bytes was not forbidden (and led to issues later)

BUGFIX: renaming a local type by pressing F2 would lead to its removal from all use sites

BUGFIX: repeatable comments for structure members were not printed when using data cross-refences instead of structure offset operands

BUGFIX: sdk: qdetach_tty()/qcontrol_tty() were leaving /dev/tty open in some cases

BUGFIX: searching for all occurrences of a byte sequence would not work without an open disassembly view

BUGFIX: TIL: layout of _TEB and _PEB structures was not correct in mssdk_win7 and later .til files.

BUGFIX: Tricore: processor module could incorrectly detect function arguments passed on stack ([sp]0 was not handled)

BUGFIX: Tricore: struct offset with selection command did not work for this processor module

BUGFIX: try block lines could be missing when reopening the IDB

BUGFIX: type parser could misbehave with fully-qualified destructor names in class definitions

BUGFIX: types: creating a c++ structure with a __vftable member in the struct view was not marking the structure as having vftable; only doing so from local types was working

BUGFIX: ui/qt: during auto-analysis, typing in the quick filter (e.g., in the 'Functions window') could result in loss of certain characters

BUGFIX: ui/qt: Hiding columns when in 'folders' mode, wouldn't work

BUGFIX: ui/qt: if entries in the "Structures" or "Enums" widgets were sorted, scrolling by using the scrollbar would jump over some entries

BUGFIX: ui/qt: on Linux, IDA could crash if some initialization failed, and IDA's main window was moved to another screen before exiting

BUGFIX: ui/qt: on OSX, IDA could appear to hang during debugging

BUGFIX: ui/qt: opening certain views (e.g., "Function calls") through the "Quick view" (Ctrl+1) could fail

BUGFIX: ui/qt: Performing undo with souce-level breakpoints defined could cause IDA to INTERR

BUGFIX: ui/qt: renaming folders in the "Local types", would show the editor on the wrong cell (in the 'Name' column, even though the folder name is in first column, named 'Ordinal'.)

BUGFIX: ui/qt: the "Command palette" could refuse to keep the user selection, making it hard to use

BUGFIX: ui/qt: the decompiler action "Jump to local type" could fail to select the proper type when the "Local types" view was sorted

BUGFIX: ui/qt: using 'Save as...' could result in an unwanted additional entry appearing in the "recent files" section of the 'File' menu

BUGFIX: ui/qt: using set_dock_pos() with no target dock and DP_SZHINT, would ignore the size hints

BUGFIX: ui/qt: when in folders mode, fast jumping by row number wouldn't work

BUGFIX: ui/qt: when searching for text in sorted folders views, IDA could loop endlessly

BUGFIX: ui/qt: while debugging, drag & dropping an unsynchronized & invisible "Pseudocode-A" tab could crash IDA

BUGFIX: ui/txt: certain commands (close, tile, cascade, ...) could trigger INTERR 49589

BUGFIX: ui/txt: it was impossible to "Import" snippets in the 'Script snippets' dialog

BUGFIX: ui/txt: opening a hex view in idat would result in a crash

BUGFIX: UI: "fast searches" in a folder view, could cause IDA to freeze, or crash in certain cases

BUGFIX: UI: a long, unbreakable line in the "Output window" would cause other long (but breakable) lines to not be laid out according to the viewport size, and thus require scrolling

BUGFIX: UI: an error message on debug start would show connection string with erronously appended port number when using WinDbg debugger

BUGFIX: UI: calling delete_menu() could cause IDA to crash at exit-time

BUGFIX: UI: choosers starting in "folder" mode, might not have the user-desired sizes for columns

BUGFIX: UI: debugger stack view could display values with wrong bitness (e.g. 32-bit values for 64-bit programs)

BUGFIX: UI: depending on the selected font, register views could truncate representation of long values

BUGFIX: UI: F1 in the 'Functions' window now shows the correct help topic

BUGFIX: UI: folder lists in various views could become empty if undo was used after saving the database

BUGFIX: UI: Hex View in databases using certain encodings (typically UTF-8), could show a glitch in the rendering of 'combining' unicode codepoints

BUGFIX: UI: IDA could crash when stopping debugging, if certain manipulations were performed on the 'Functions window'.

BUGFIX: UI: IDA on Windows could show a warning "The operation completed successfully." when checking for updates using an up-to-date build.

BUGFIX: UI: in "Structures" and "Enums", creating a new type when the tree selection is not a folder, would create the type at the toplevel instead of the current one

BUGFIX: UI: in cases where the address space is very fragmented, zooming in the navigation band could lead to incorrect positioning

BUGFIX: UI: in folders view, triggering a rename, but not actually renaming (by e.g., leaving the name untouched, or clicking somewhere else), would cause an annoying message in the "Output window".

BUGFIX: UI: in the "Output window", if a long line had to be broken up into multiple 'physical' lines, clicking in the middle of one of those physical lines would place the cursor to its beginning

BUGFIX: UI: in the "Structures" and "Enums" widget, jumping to a structure or enum that's currently not selected, could either fail, or cause the companion tree to be out-of-sync

BUGFIX: UI: in the "Structures" or "Enums" widget, selecting a folder containing items, and deleting that folder, wouldn't properly update the listing contents

BUGFIX: UI: in the "Structures" or "Enums" widgets, the listing could be missing types after an undo operation

BUGFIX: UI: invoking "Xref graph" commands could produce "Wrong specification" warning intead of showing the graph

BUGFIX: UI: it was impossible to expand the hints shown in the "IDA View-A", when the cursor was positioned on an 'XREF'.

BUGFIX: UI: it was impossible to turn columns off/on in tabular views when they are in folders mode

BUGFIX: UI: filtering folders-enabled views, should hide the folders that don't have any content

BUGFIX: UI: pressing Enter in a dirtree widget would erroneously edit the item on macOS.

BUGFIX: UI: when exporting data as a "String literal", IDA could fail to properly decode text for IDBs that use UTF-8 as internal encoding

BUGFIX: UI: when using dark theme on macOS and linux, text within combobox menus could be clipped

BUGFIX: UI: when using dark theme on macOS, selected items in tree views could be colored incorrectly.

BUGFIX: UI: list of local types could change after enabling folders

BUGFIX: UI: opening an ida_kernwin.PluginForm at a specific position could fail

BUGFIX: UI: quick searching (i.e., by simply typing a string) in tabular/tree views might not yield the expected results.

BUGFIX: UI: rebasing a tree widget could cause IDA to show empty entries

BUGFIX: UI: scrolling in the navigation band could jitter with very segmented address spaces

BUGFIX: UI: searching for all occurrences of a binary data in "Hex View-1", closing "Hex View-1" and then attempting to jump, would cause IDA to crash

BUGFIX: UI: selecting a portion of an identifier for highlighting, and then searching up/down (Alt+Up/Alt+Down) for that text, would cause the entire identifier to become highlighted

BUGFIX: UI: sorting folders would only sort folders contents, but not the folders themselves

BUGFIX: UI: the 'grabbable' band for a floating window (used to dock it back) was not easy to spot

BUGFIX: UI: the "Current line" message could fail to display in some views when folders were enabled

BUGFIX: UI: the "Watch List" window wouldn't refresh when one of its items was renamed from the disassembly listing

BUGFIX: UI: the column "func#" in the Signatures list was not properly sortable as a number

BUGFIX: UI: types could be duplicated in the folder view of 'Local types' window

BUGFIX: UI: Undo wouldn't cause previously-rebased 'Imports' to get their original address back

BUGFIX: UI: using quick filtering with a regex, wouldn't highlight the part of the string that matches (as it does for lexicographical searches)

BUGFIX: UI: when folders are enabled in tabular views, 'Copy/Copy all' could fail to work as expected.

BUGFIX: UI: when folders were enabled on certain widgets, and the IDB was saved (e.g., by clicking on the 'save' icon), but then not saved again when closing, the widget would show up in no-folders mode

BUGFIX: UI: when switching to a folders-enabled chooser, the folders might fail to have focus

BUGFIX: UI: with certain fonts, the 'registers' widget could truncate the register names/values by a few pixels, making it harder to read

BUGFIX: UI: zooming in the navigation band, could lose locality

BUGFIX: Using "Jump to local type..." in a Pseudocode view's context menu wouldn't expand the tree of types to the right place (assuming the "Local types" has been toggled to be using folders.)

BUGFIX: debugger: single-stepping could be sluggish with multi-threaded apps, especially in the Mac debugger (see Debugger>Debugger options>Optimize single-stepping. it is enabled by default for Mac debugger).

BUGFIX: decompiler: "create new struct type" could generate a new struct type with forbidden characters, like <

BUGFIX: decompiler: arm64: some references to external symbols would not be resolved

BUGFIX: decompiler: assignments to members of structurs later passed by reference could be optimized away

BUGFIX: decompiler: automapping variables was too aggressive in some cases

BUGFIX: decompiler: bit-manipulation instructions 16-bit registers were emulated incorrectly (shift count is only 4 bits for them)

BUGFIX: decompiler: changing the type of a structure field would cause the loss of the __cppobj attribute

BUGFIX: decompiler: decompile() would crash if asked to decompile an unexisting function (nullptr)

BUGFIX: decompiler: decompiler could crash after using "extract func" if a pseudocode window with the deleted function info was present

BUGFIX: decompiler: fixed a crash in the recognition of magic divisions

BUGFIX: decompiler: fixed a crash on corrupted idbs

BUGFIX: decompiler: fixed an endless loop (extremely rare)

BUGFIX: decompiler: fixed interr 52194

BUGFIX: decompiler: global xref cache could become stale after a user action that was changing only the line numbers (like adding a comment)

BUGFIX: decompiler: in some cases 'split expression' had no effect

BUGFIX: decompiler: in some cases action "Reset pointer type" was not working (had no effect)

BUGFIX: decompiler: in some cases decompiler could generate a wrong post-increment/decrement operator

BUGFIX: decompiler: in some cases the decompiler would add a suffix to the user-defined names (myvar->myvara)

BUGFIX: decompiler: in some cases the user could not add variadic arguments to a function call

BUGFIX: decompiler: jumping to the pseudocode from another window (for example, from the local types) would fail to activate the window in some cases

BUGFIX: decompiler: ppc: fixed address arithmetics when subtracting addresses

BUGFIX: decompiler: recognition of inlined memset() was too aggressive in some cases (a xor loop could be converted to memset)

BUGFIX: decompiler: renaming a structure field would cause the loss of the __cppobj attribute

BUGFIX: decompiler: shifted pointers with negative offsets were not always applicable

BUGFIX: decompiler: some assignments could not be split

BUGFIX: decompiler: some xrefs to enum members would be missed by Ctrl-Alt-X

BUGFIX: decompiler: stale cached pseudocode was not refreshed in some cases

BUGFIX: decompiler: the decompiler could crash when displaying the global xref list if the cache was stale

BUGFIX: decompiler: trying to rename a variable as "@@x" would lead to a fatal error

```
BUGFIX: decompiler: while(2){switch...} could be decompiled
incorrectly in some cases

BUGFIX: decompiler: wrmsr instruction could be decompiled wrongly
(value of edx was unused)

BUGFIX: decompiler: x87 fscale instruction was decompiled
incorrectly

BUGFIX: windbg: segment bitness could be determined incorrectly
```