IDA 9.0

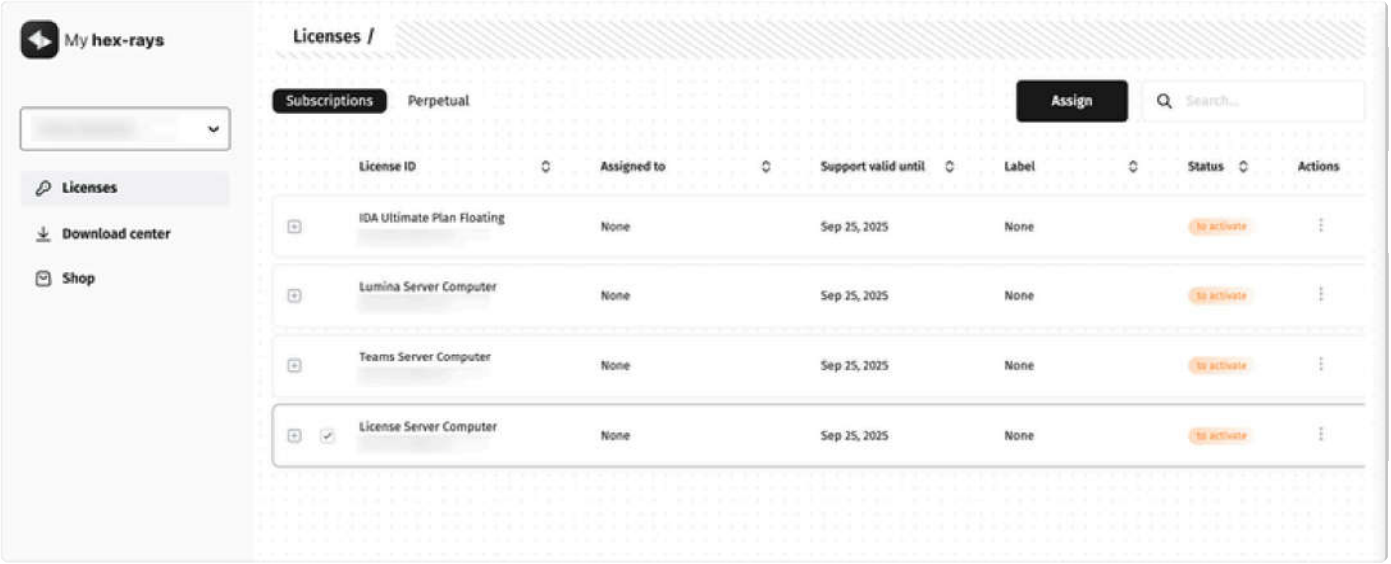
IDA 9.0.240925 ***September 30, 2024***

Looking to try IDA 9.0? Find out how to upgrade now and request your IDA 9.0 trial.

IDA 9.0 Highlights

Licensing changes

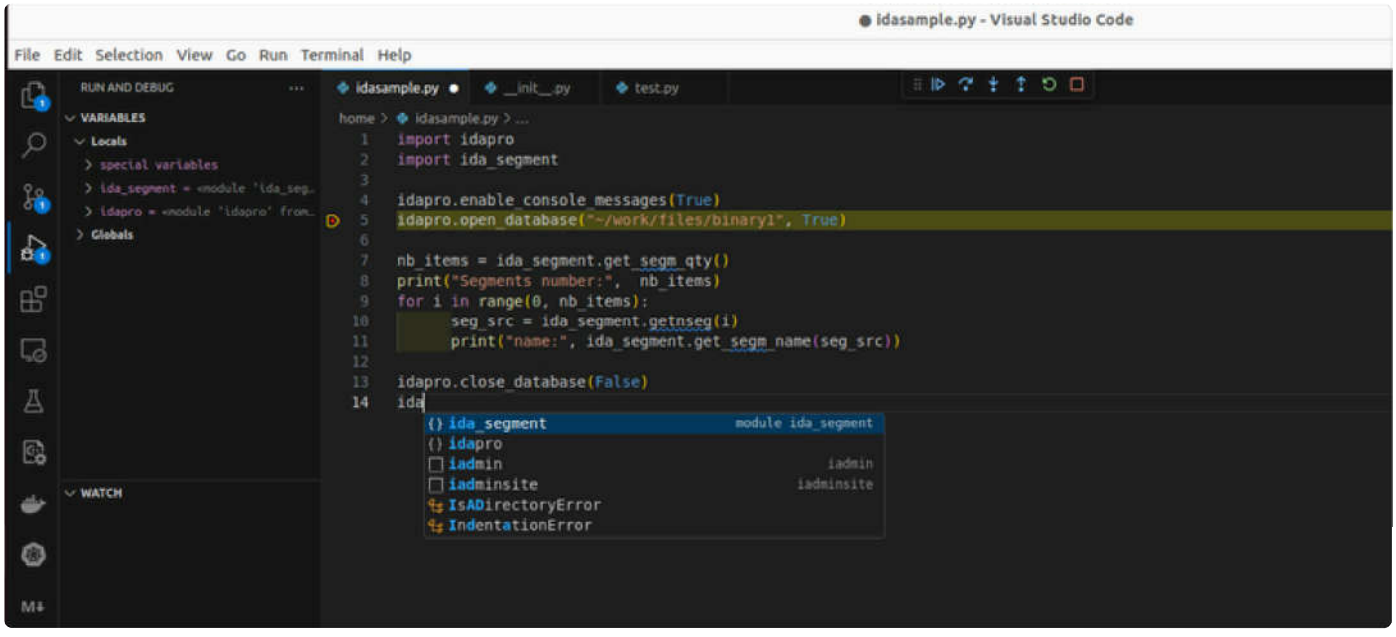
- Licenses are no longer bound to a specific platform. After buying one IDA license, it can be used on all supported platforms (Windows/Linux/macOS).
- License packs with various decompilers are available
- IDA Home 68K is retired and replaced by the new IDA Home RISCV (with cloud decompiler!)
- IDA Teams and Private Lumina functionality are available as options and can be used with standard IDA Pro
- IDA Teams users count is no longer limited by the seat count — only the concurrent usage is enforced according to IDA's license type
- A custom Hex-Rays licensing server replaces the FlexNet licensing server for floating licenses



Customer portal - licenses view

Headless processing with idalib

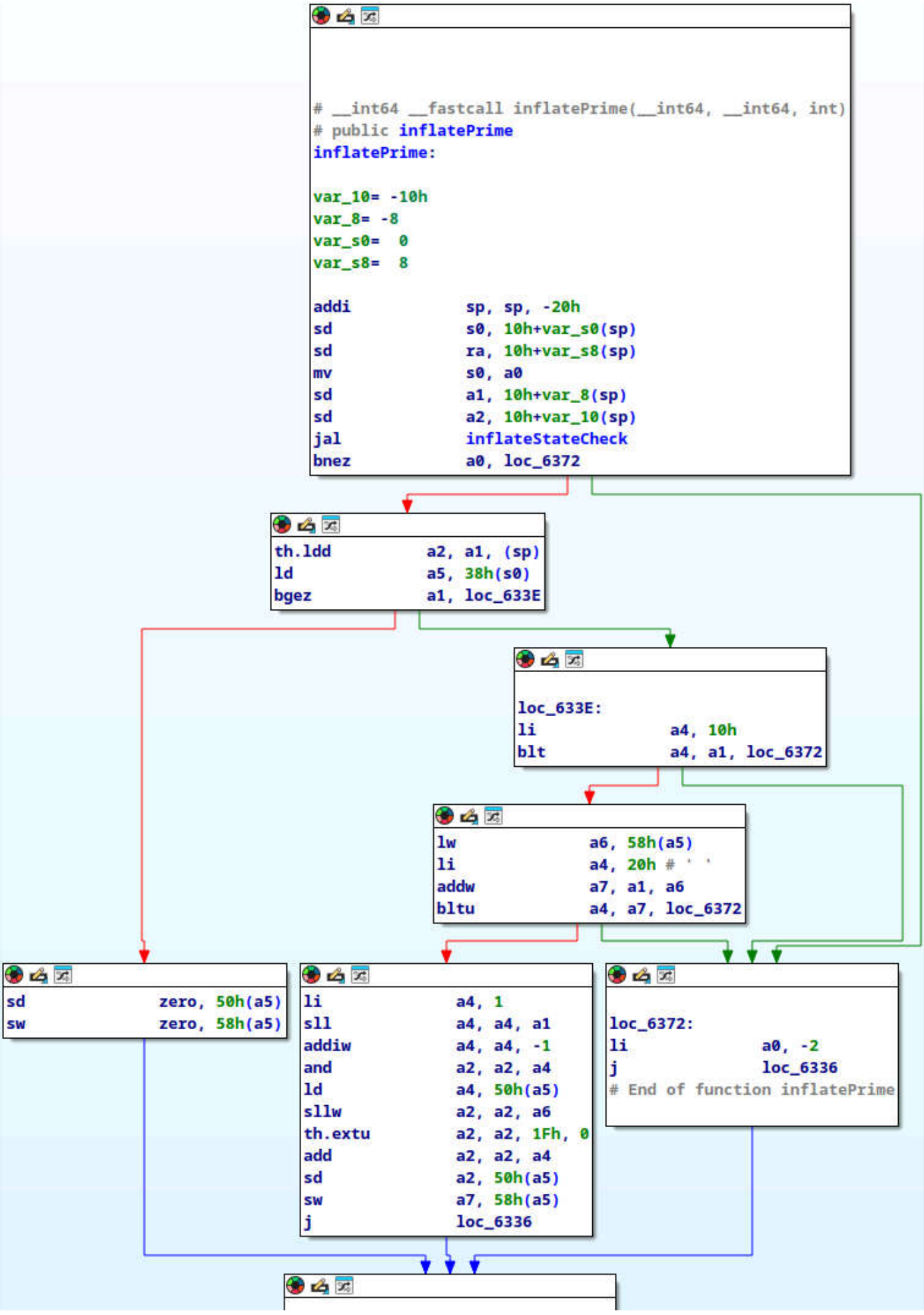
- With idalib, both the C++ and Python APIs can be used from outside IDA to form standalone applications. The resulting program or script doesn't have to be loaded inside IDA, but rather IDA's engine is used inside your application.
- This makes developing against the IDA API much easier — if configured correctly, you get auto-completion and debugging in your favorite C++/Python IDE
- NO RPC or IPC to an external IDA process means you get a native speed of execution
- Almost any Python can be used: official CPython, Anaconda, Homebrew, etc.



Screenshot of idalib

New RISC-V Decompiler and Disassembler Extensions

- New decompilers targeting 32- and 64-bit RISC-V code (HEXRV and HEXRV64) are now available
- We extended the RISC-V processor module to support T-Head extension instructions (used in Xuantie and Allwinner processors)



__int64 __fastcall inflatePrime(__int64, __int64, int)
public inflatePrime
inflatePrime:
var_10= -10h
var_8= -8
var_s0= 0
var_s8= 8
addi sp, sp, -20h
sd s0, 10h+var_s0(sp)
sd ra, 10h+var_s8(sp)
mv s0, a0
sd a1, 10h+var_8(sp)
sd a2, 10h+var_10(sp)
jal inflateStateCheck
bnez a0, loc_6372
th.ldd a2, a1, (sp)
ld a5, 38h(s0)
bgez a1, loc_633E
loc_633E:
li a4, 10h
blt a4, a1, loc_6372
lw a6, 58h(a5)
li a4, 20h # ' '
addw a7, a1, a6
bltu a4, a7, loc_6372
sd zero, 50h(a5)
sw zero, 58h(a5)
li a4, 1
sll a4, a4, a1
addiw a4, a4, -1
and a2, a2, a4
ld a4, 50h(a5)
sllw a2, a2, a6
th.extu a2, a2, 1Fh, 0
add a2, a2, a4
sd a2, 50h(a5)
sw a7, 58h(a5)
j loc_6336
loc_6372:
li a0, -2
j loc_6336
End of function inflatePrime

```
loc_6336:
ld              ra, 10h+var_s8(sp)
ld              s0, 10h+var_s0(sp)
addi            sp, sp, 20h # ' '
ret
```

Screenshot of RISC-V disassembly

IDA View-A | Pseudocode-A

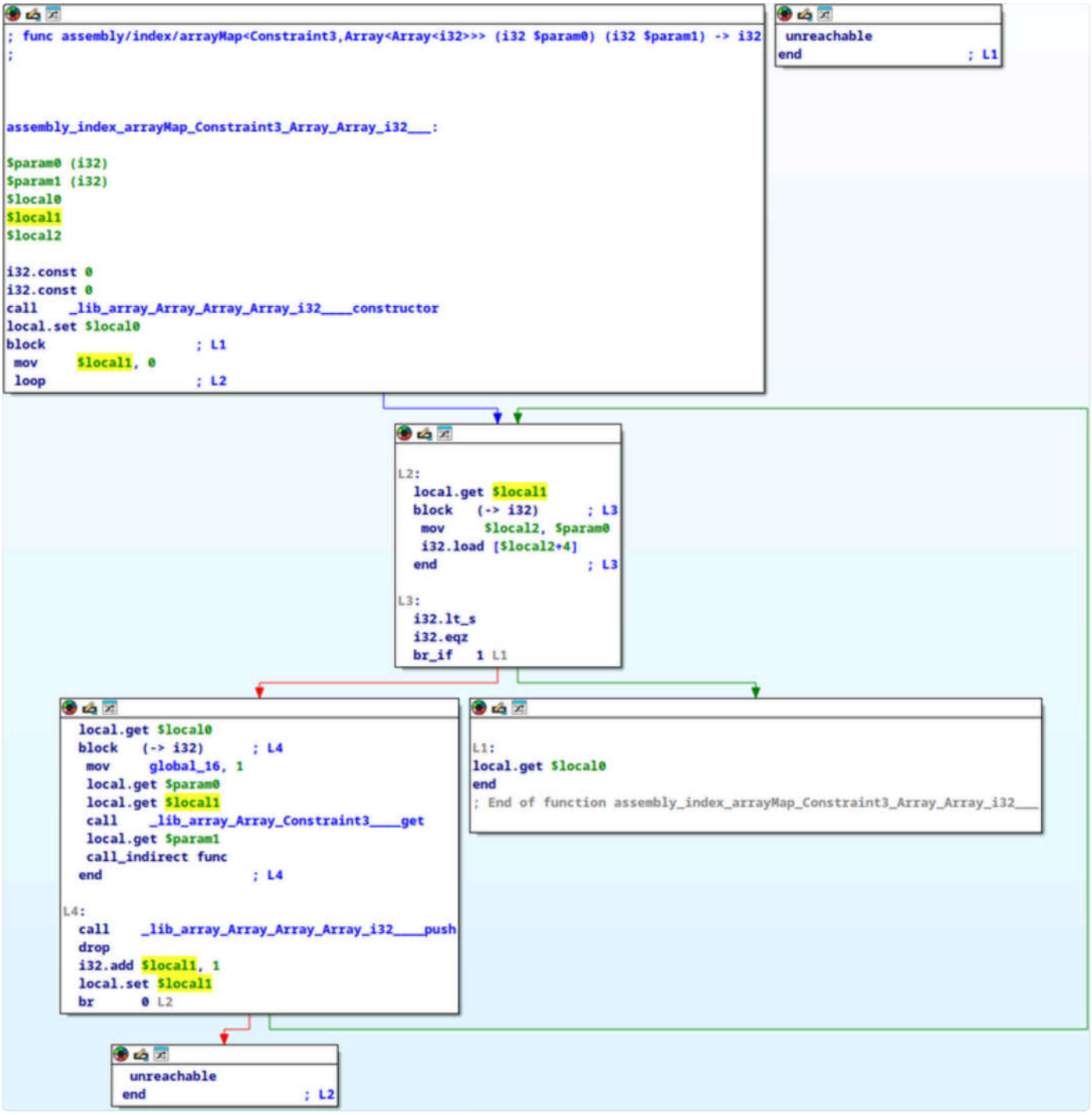
```
__int64 __fastcall crc32_combine_(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 v5; // s1
  __int64 v6; // a5
  __int64 v7; // a4
  _BYTE v9[256]; // [sp+0h] [-208h] BYREF
  _QWORD v10[33]; // [sp+100h] [-108h] BYREF

  if ( a3 > 0 )
  {
    v10[0] = 0xEDB88320LL;
    v5 = a3;
    v6 = 1LL;
    v7 = 1LL;
    do
    {
      v10[v6++] = v7;
      v7 *= 2LL;
    }
    while ( v6 != 32 );
    gf2_matrix_square(v9, v10);
    gf2_matrix_square(v10, v9);
    do
    {
      gf2_matrix_square(v9, v10);
      if ( (v5 & 1) != 0 )
        a1 = gf2_matrix_times(v9, a1);
      if ( !(v5 >> 1) )
        break;
      gf2_matrix_square(v10, v9);
      if ( ((v5 >> 1) & 1) != 0 )
        a1 = gf2_matrix_times(v10, a1);
      v5 >>= 2;
    }
    while ( v5 );
    return a1 ^ a2;
  }
  return a1;
}
```

Screenshot of RISC-V decompilation

WASM Disassembler and File Format Loader

- With many apps shifting to client-side browser applications, we saw the need for a new disassembler for Web Assembly (WASM)

- WASM code is embedded into its own binary file format hence we also ship a file loader that decodes the WASM file format



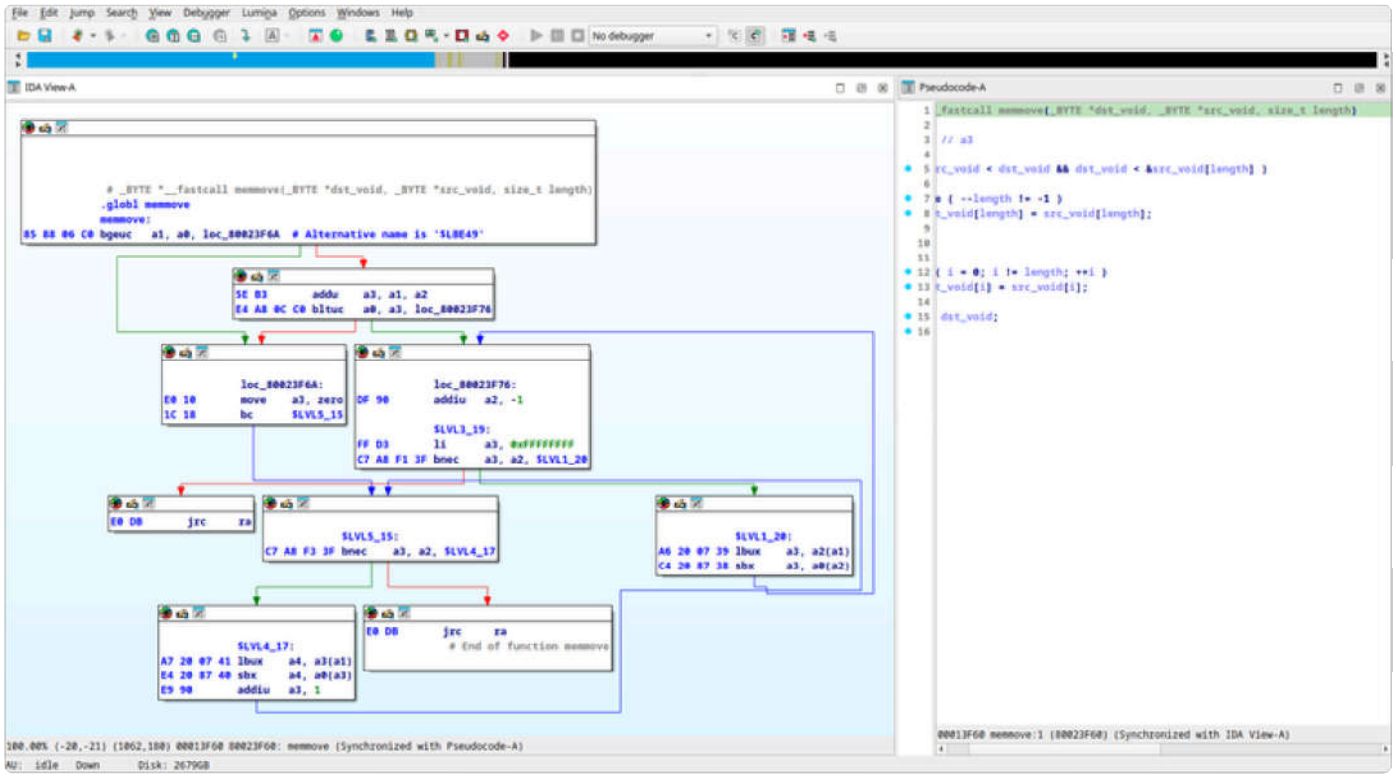
Screenshot of WASM disassembly

nanoMIPS support

- Both the MIPS disassembler and decompiler now support nanoMIPS instructions
- Despite the name, it's not a simple extension of the MIPS ISA but a completely new encoding of the existing MIPS instructions and addition of new ones, as well as a brand

new calling convenion

- nanoMIPS support is included in the MIPS decompiler (HEXMIPS), there is no need for an extra license
- Firmware compiled for nanoMIPS often ships in md1rom format, which is why we added md1rom file loader to IDA (including parsing and applying of debug symbols, if available)

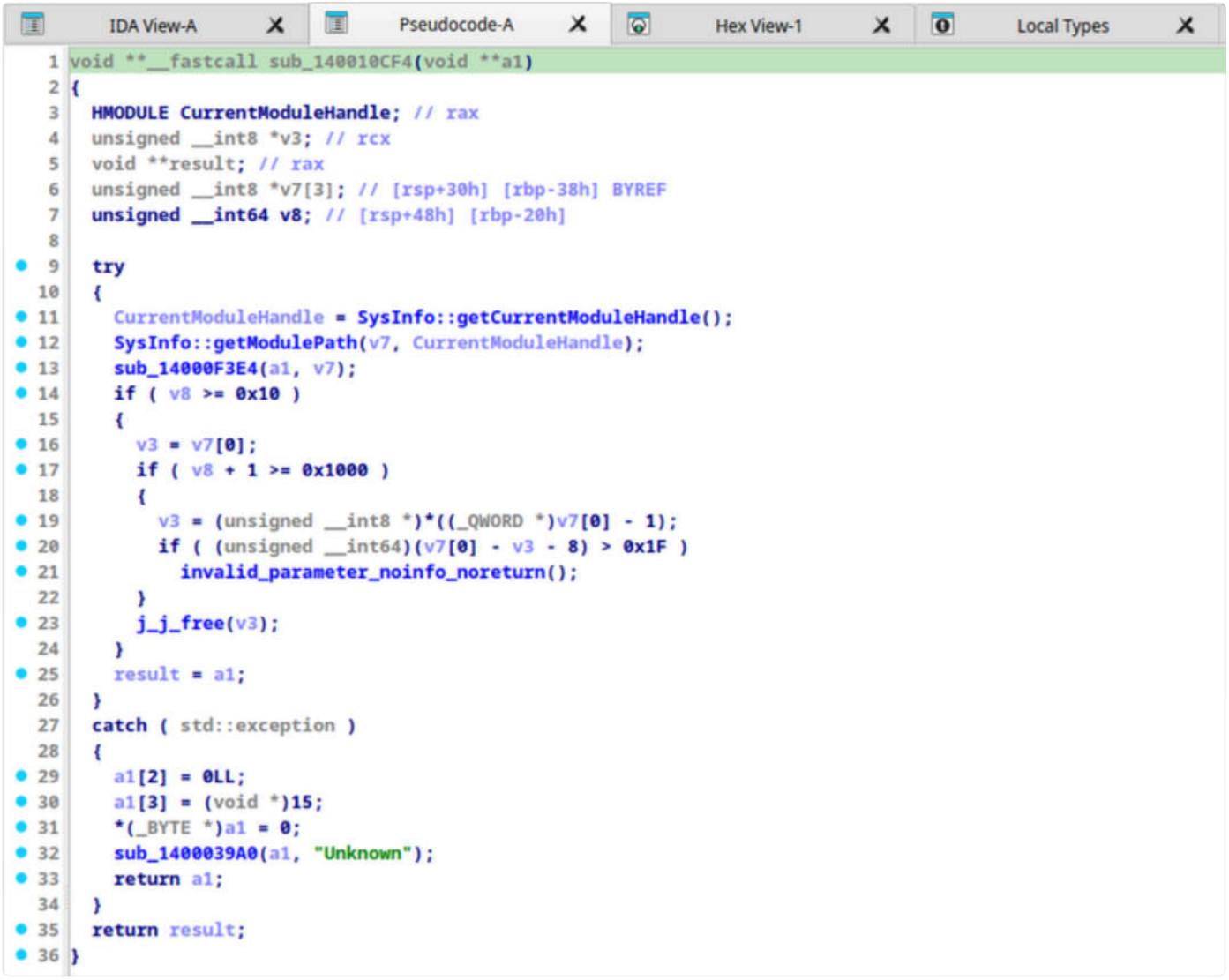


Screenshot of nanoMIPS disassembly/decompilation

C++ Exceptions Support in the Decompiler

hex-rays / Docs

VC++.

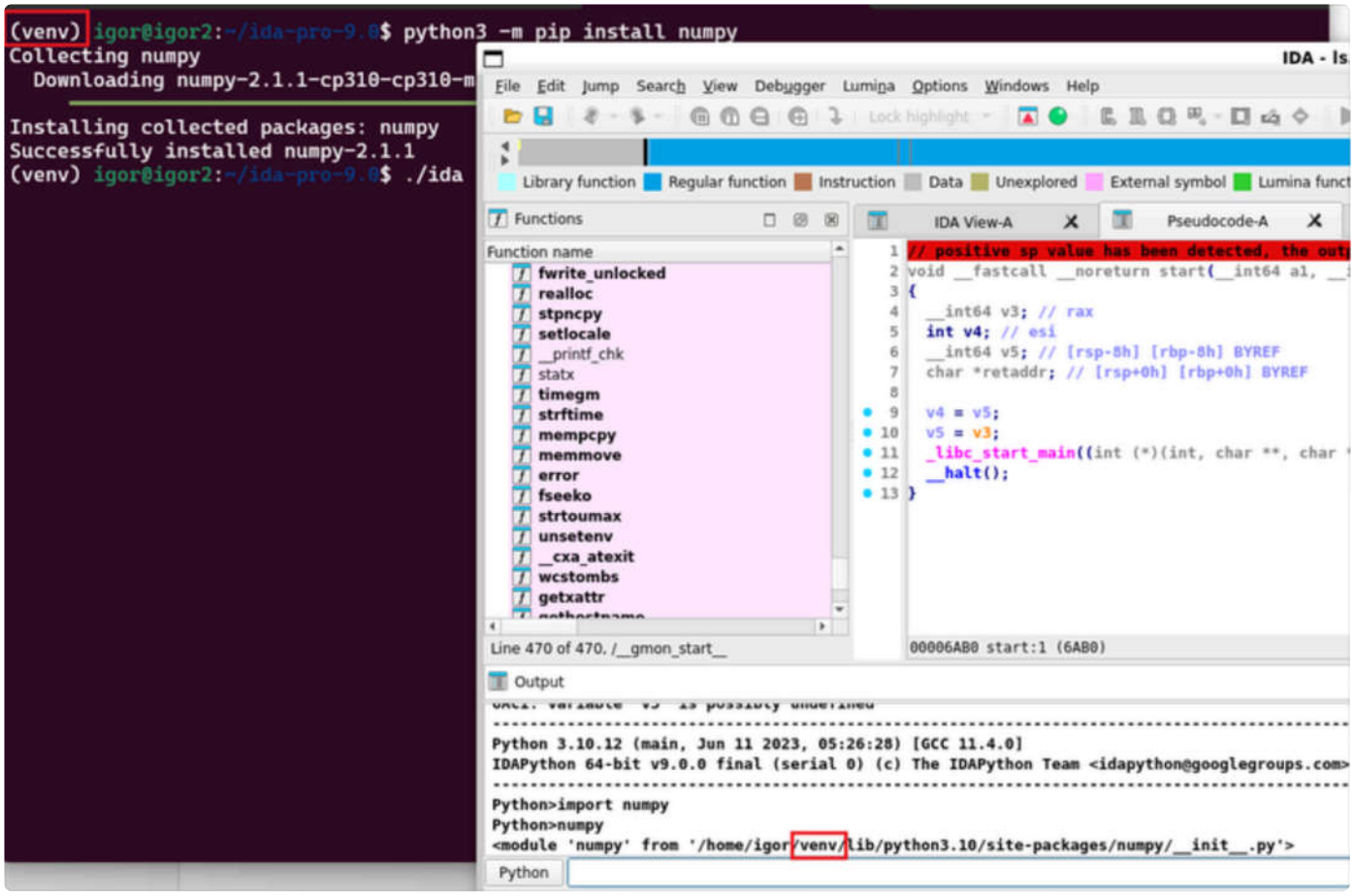


Screenshot of C++ try/catch blocks in pseudocode

IDAPython Improvements

- Most IDAPython APIs now have type annotations, making the API less obstructive to use.
- Python virtual environments (venvs) are now supported - simply run IDA from an activated virtual environment and it will pick up locally installed modules
- Objects returned in the Python API are properly zero-initialized.
- `idapyswitch` can now be used with read-only IDA installations (nothing is changed in the installation directory when picking a different Python version/install)
- Auto-completion in IDA's CLI now disregards `__magic_methods__` and auto-generated SWIG methods, reducing noise and helping to find a particular function faster
-

Auto-completing a method call shows its prototype with type annotaions and docstring (if available) in a pop-up hint

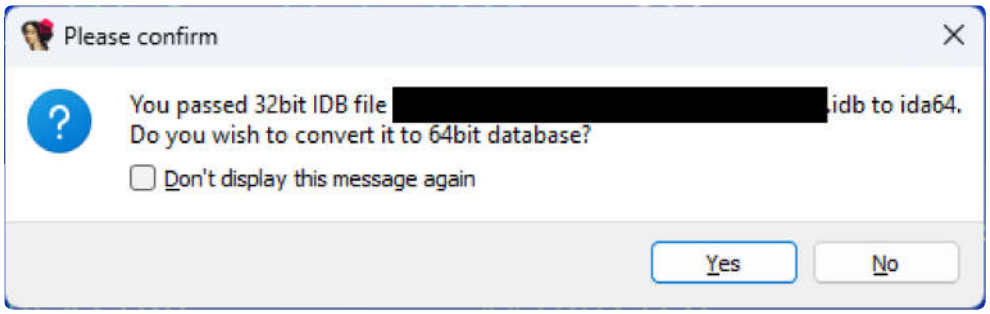


Screenshot of IDA run from venv

Screenshot of IDAPython completion hint

No more IDA32

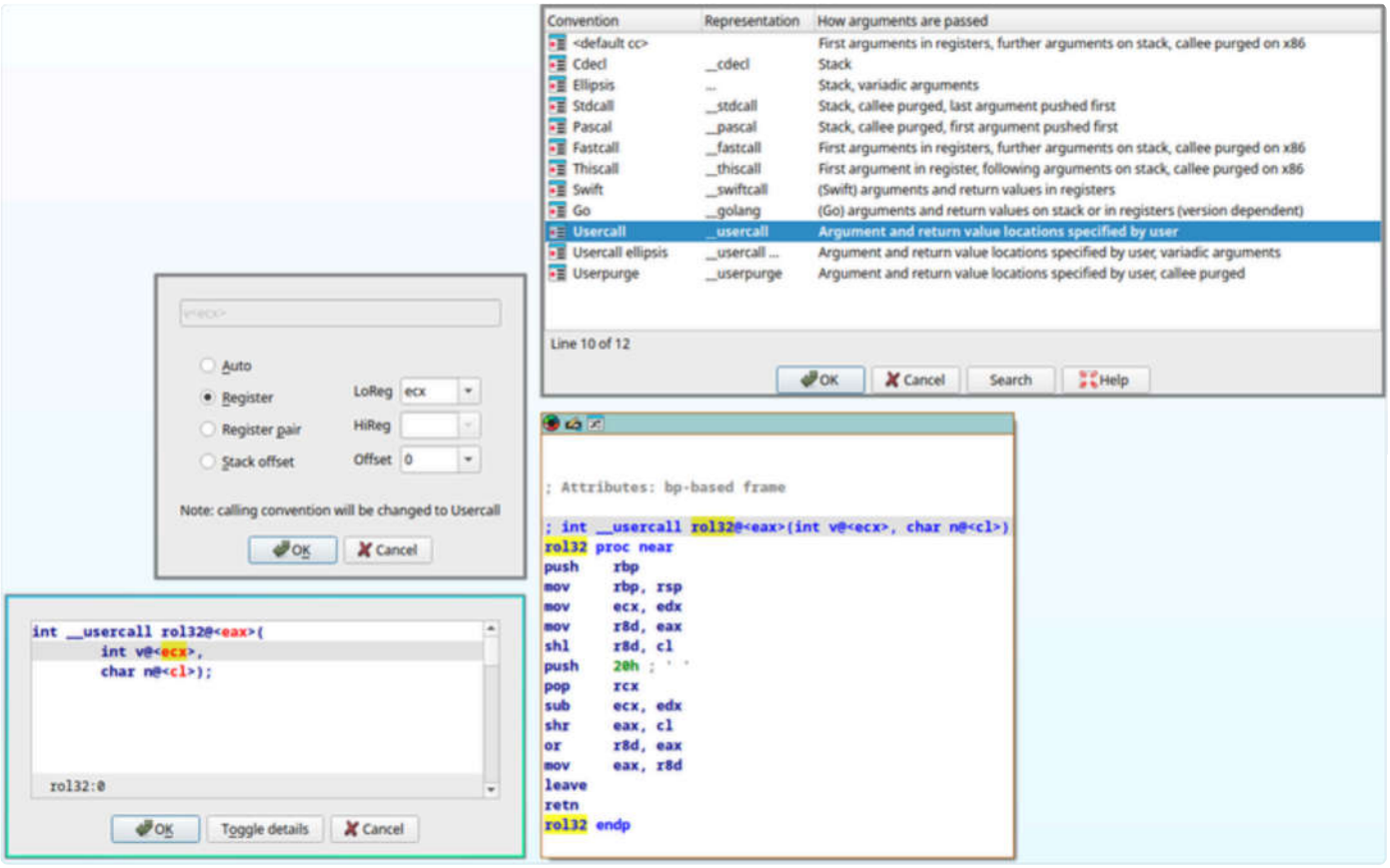
- We deprecated IDA32 a few versions ago. With IDA 9.0, just one IDA binary handles both 32- and 64-bit code.
- The number of installed executable files is cut in half
- An easier life for native plugin maintainers since only one version (`__EA64__=1`) needs to be maintained
- The conversion of legacy IDB into the I64 file format is transparent and automatically performed by IDA



Screenshot of idb conversion prompt

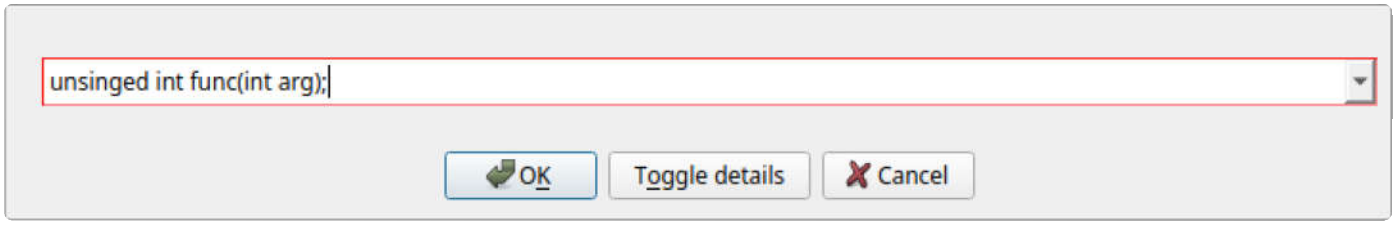
UI Improvements

- The legacy Enums and Structures views are removed entirely and replaced by the Local Types.
- This also means that `struct.hpp` and `enum.hpp` and their Python counterparts `ida_struct` and `ida_enum` disappear from the API. Replacement functionality for both headers/modules is now located (mostly) in `typeinf.hpp` / `ida_typeinf`. A porting guide is available.
- It is now possible to specify fixed size for structures and to enable field packing easily
- The function prototype editor (aka `Y` shortcut on a function name) now can toggle between the classic free-text one-line editor and a new multi-line editor featuring the usual shortcuts and controls.
- At the same time, we added basic support for UI-based editing of argument locations, to make our custom `__usercall` syntax less of a hassle to remember.



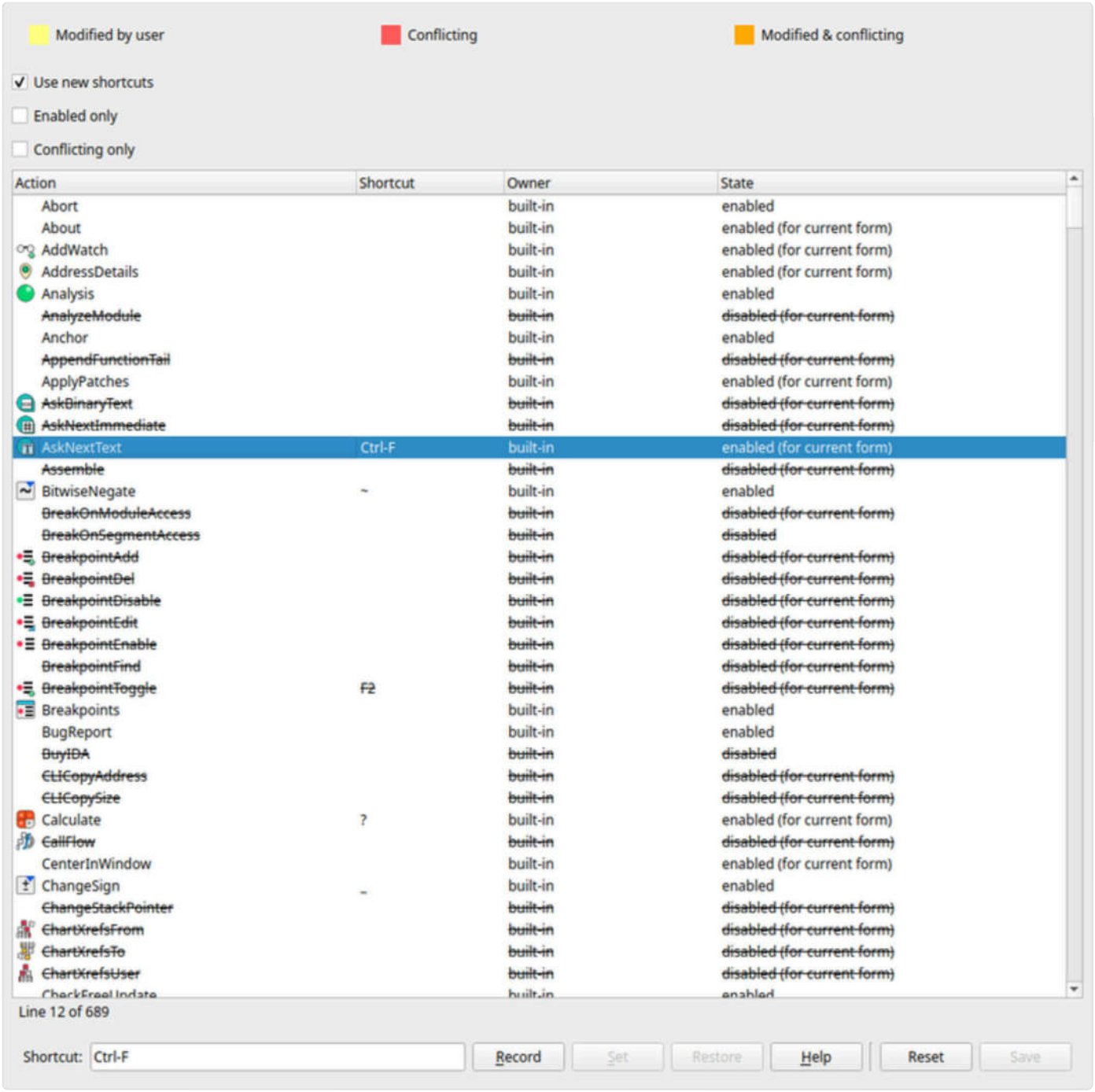
Screenshot of new prototype editor

- The basic function prototype editor now indicates invalid prototypes via a red rectangle while typing



Screenshot of new prototype editor

- A refreshed set of shortcuts that better matches the modern OS conventions can now be selected instead of the traditional shortcuts



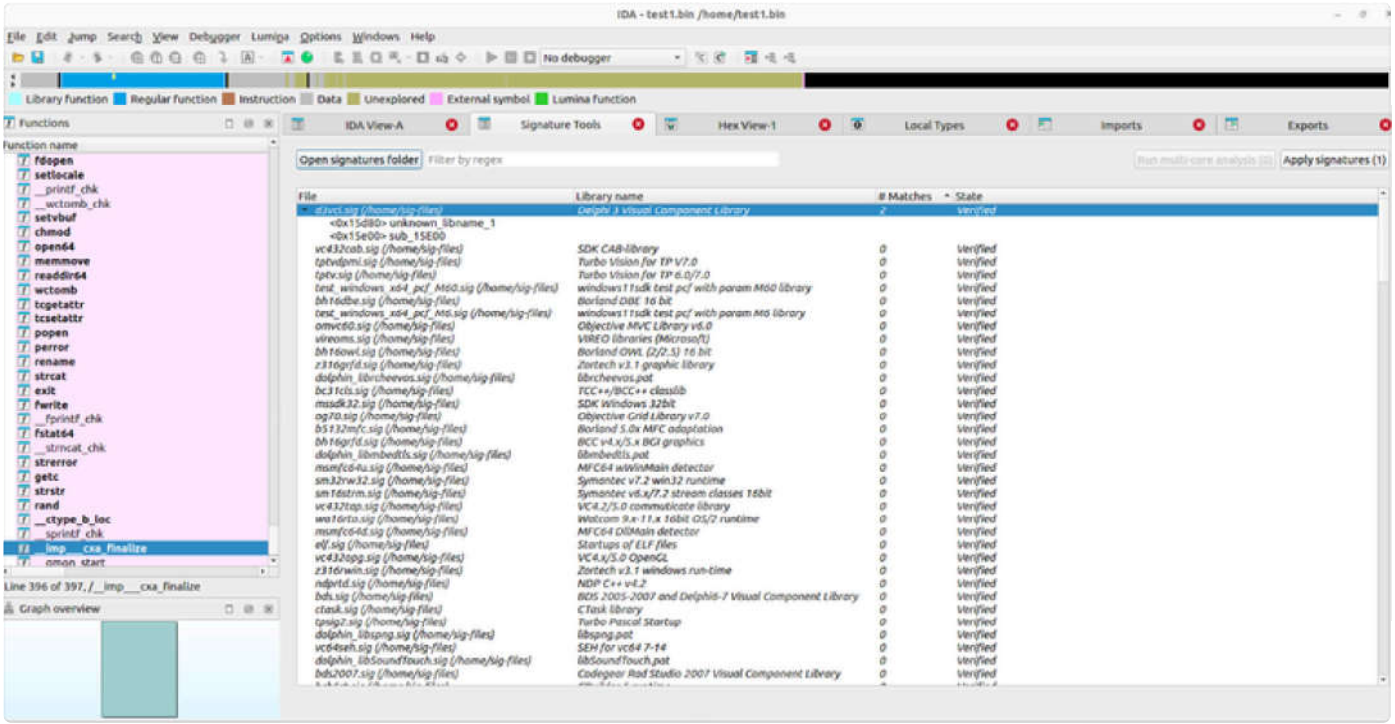
Screenshot of new shortcuts

FLIRT Updates

- We massively updated, modernized and extended the number of FLIRT signatures available for use with IDA
- Signatures for modern languages like Golang and Rust, as well as updates for classic compilers (MSVC for Windows, GCC for Linux) are made available as a separate download due to their size
-

- These signatures are generated and updated automatically with every new release of the compiler/language/library
- A new plugin (FLIRT Manager) allows easy application of multiple signatures to the database and lets you see which one gives the best results

```
- Golang:
  - Versions: stable versions from 1.10.0 to 1.23
  - Operating Systems: Linux, Windows, MacOS
  - Architectures: arm64 (Windows, Linux, MacOS), arm (Windows, Linux, MacOS), x
- C/C++
  - Windows (MSVC):
    - Architectures: arm, arm64, i386, amd64
    - Packages: ATL, CRT, MFC, Windows SDK 10, Windows SDK 11
  - Linux:
    - Distribution: Ubuntu & Debian
    - Architectures: i386, amd64, arm64, armhf, armel, arm, s390x, mips64el, mip
    - Packages: libc6, libselinux1, libpcre2, libidn2, libssl, zlib1g, lib32z1,
- Rust
  - Versions: 1.77 to 1.81
  - Architectures: arm64, arm, x86, x86-64
  - Operating Systems: Linux, Windows, MacOS
  - Compilers: GCC, LLVM, MSVC
```



Screenshot of FLIRT Manager

Metadata Descriptors for Plugins

- `ida-plugin.json` now offers a standardized entrypoint for plugins. This enables plugin authors to follow their own plugin directory structure, all they need to do is point IDA to the main plugin entry point. To maintain backward compatibility, IDA will keep loading plugins in the legacy way for a couple of releases.

  With the following directory structure:

```
plugins
└── ida_greeter
    ├── ida-plugin.json
    └── main.py
```

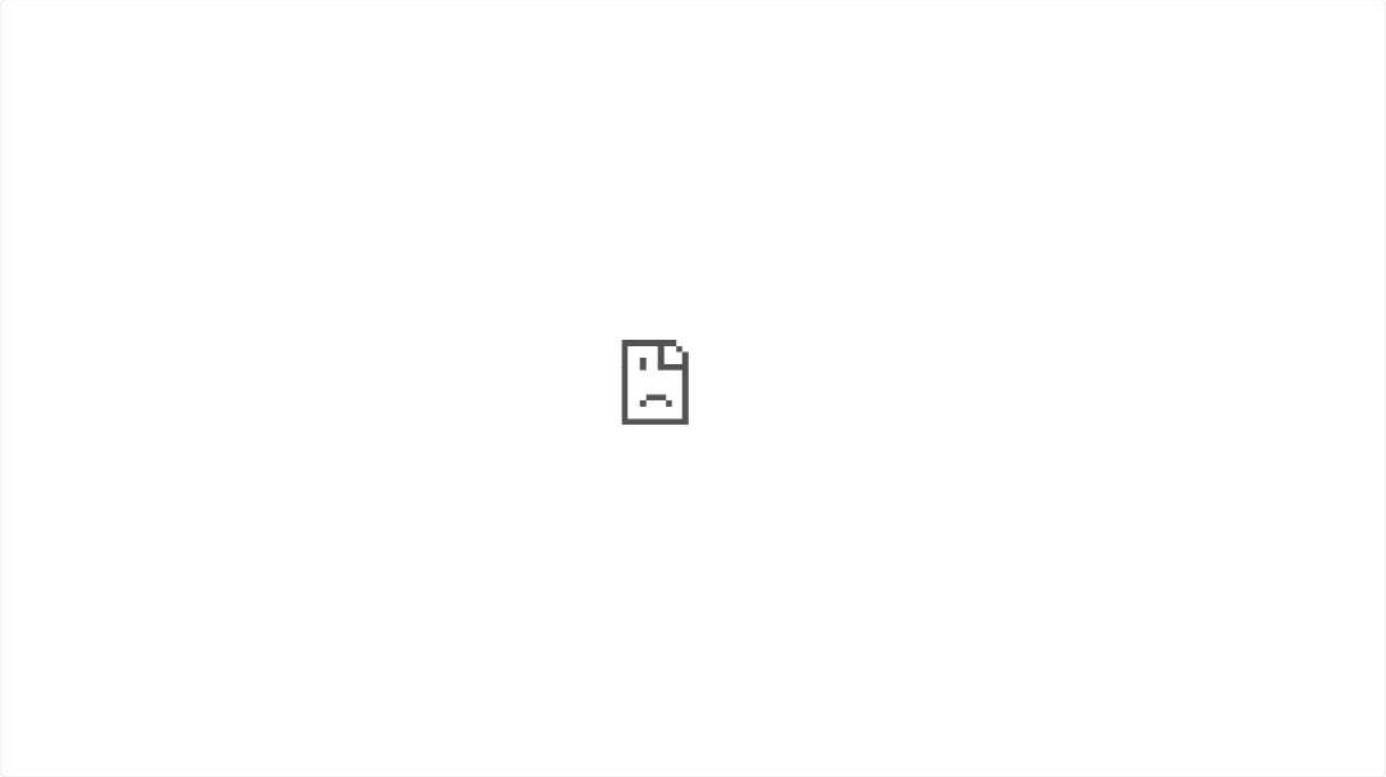
A possible `ida-plugin.json` could look as follows:

```
{
  "IDAMetadataDescriptorVersion": 1,
  "plugin" :
  {
    "name" : "greeter",
    "entryPoint" : "main.py"
  }
}
```

- This approach allows for easy management of plugin's resources and bundled dependencies

Watch what's new in IDA 9.0

Curious about the new IDA? Watch the feature overview on the All Things IDA channel.



Courtesy of Elias Bachaalany (@allthingsida)

Full list of changes and new features:

Processor modules

- 68K: added typical code start sequences
- ARM: improved detection of targets of indirect jump instructions
- ARM: improved prolog analysis to recognize and mark calls to `chkstk_darwin`
- AVR: updated missing bit definitions for ATmega640
- MIPS: support for NanoMIPS instruction set
- RISCV: added support for legacy instruction sfence.vm
- RISCV: added support for T-Head custom instructions
- RISCV: fixed the frame analysis when s0 (FP) is saved
- wasm: new processor module (Web Assembly)
- RH850: added new instructions supported by RH850G4MH core (SIMD, FXU, etc.)
- V850/RH850: convert two-instruction loads and stores into one macroinstruction

File formats

- ELF: added support for nanoMIPS
- ELF: ARM64: added support for `R_AARCH64_P32_TLS_TPREL` relocation type, used by ILP32
- ELF: RISCV: added suport for R_RISCV_ALIGN relocation type
- md1img: loader for Mediatek modem firmware images (nanoMIPS and MIPS16e2)
- MACHO: support `__chain_starts` format 5 (`DYLD_CHAINED_PTR_32_FIRMWARE`)
- MACHO: handle iOS18 DSC with zero-sized `__OBJC_RO` segment in libobjc
- wasm: new file loader for Web Asembly modules

FLIRT / TILS / IDS

- FLAIR: PCF: added support for ARM64 COFF files
- FLAIR: PELF: proper handling of ELF32 for AArch64 (ILP32)
- FLAIR: PELF: added support for most common relocation types for MIPS, MIPS64, ARM, AARCH64, PPC, PPC64, PARISC, SPARC, M86K

Standard plugins

- eh_parse: skip leading and trailing zero entries in x64 `.pdata` for PE files (real binaries have them); improve recognition of exception dispatcher functions in debug builds
- eh_parse: x64 exception handlers are now proper standalone functions and not function chunks
- eh34: new plugin to handle c++ exceptions for the binaries built by msvc x64
- ida_feeds: new plugin and standalone script for mass application of FLIRT signatures
- makesig: add run() method which can be used to generate .sig (or just pat) from the database in batch mode
- pdb: added an option to only load names (useful with large PDBs when you don't need types)
- pdb: allow user to choose what to load for a module (types and/or names) during debugging

Kernel/Misc

- goodname.cfg: improve simplification of MSVC STL classes
- kernel: c/c++ keywords are now forbidden as struct fields
- kernel: support for ida-plugin.json
- kernel: improved strlit detection (short ones were converted to data items)
- kernel: improved recognition of noret functions which call other noret functions indirectly
- noret.cfg: added terminate, std_terminate to the list of non-returning functions
- installer: macOS: install all contents into a single `.app` bundle
- licensing: replaced FlexNet licensing server by custom Hex-Rays licensing server (floating licenses only)

Scripting & SDK

- IDAPython: added `find_binary` and `find_string`
- IDAPython: added detection of virtual environments (venv)
- IDAPython: added more pointer wrappers for integer types defined in pro.h
- IDAPython: added `cli_t.OnFindCompletions` replacing `cli_t.OnCompleteLine`
- IDAPython: idapyswitch can now be used with read-only IDA installations
- IDAPython: idapyswitch can now detect recent homebrew versions on macOS
- IDAPython: Removed `__magic_methods__` from CLI auto completion
- IDAPython: zero-initialize C++ objects exposed in the Python API
- IDAPython: simplify directory structure (got rid of '3', and 'ida_32|64' became 'lib-dynload')
- IDAPython: `loader_input_t.read()` should return an empty `bytes` object upon read error, not `None`
- SDK: added Visual Studio templates for plugins and loaders
- SDK: added `get_last_widget(mask)`
- SDK: added `FUNC_UNWIND` / `FUNC_CATCH` function flags to mark exception handlers, they will be ignored in decompilation
-

- SDK: added `pipe_process()` to launch a process and establish a 2-way communication with it
- SDK: added `qlist::splice()`
- sdk: extended `cli_t` interface to allow retrieving function prototypes and docstrings on auto completion
- sdk: introduced flags `IRI_...` to be used in `is_ret_insn()`, `ev_is_ret_insn` instead of `bool strict`
- SDK: moved `node_ordering_t` to `gdl.hpp`
- SDK: package decompiler's interface (hexrays.hpp) and samples as part of the SDK instead of inside IDA
- SDK: published basic undo interface (create undo point, undo, redo)
- SDK: renamed `abstract_graph_t` → `drawable_graph_t`; `mutable_graph_t` → `interactive_graph_t`

UI

- UI: added an option to retain structure size (Fixed size structs)
- UI: added "pack fields" checkbox to control gaps between fields for structs
- UI: added syntax highlighting for user-defined types in the freetext editor
- UI: command palette: fix wrong reports about "command failed"
- ui: graphs: do not display a prompt when there's only one choice for jumping to a parent/child node
- UI: handle export/import of Local types to IDC is in a more flexible way. User is able to select the different policies, for example: load the types and skip the equal.
- UI: if IDA already has a file open, File > Open or dropping a file on its window opens it in a new IDA instance (configurable via `OPEN_IDB_IN_NEW_WINDOW` in `idagui.cfg`)
- UI: it is now possible to inspect contents of base type libraries, by double-clicking on them in the "Type libraries" view
- UI: introduced a new set of keyboard shortcuts better aligned with modern OS conventions
- UI: got rid of "Structs" and "Enums" widgets
- UI: new shortcuts: Alt- (and CMD-) to jump to a window
- UI: enabled Wayland support on Linux
-

- HVUI: added a new action "Convert IDB"; it converts the idb and replaces it with i64. bulk operation is also possible

Decompilers

- decompiler: riscv: added RV32 and RV64 decompilers
- decompiler: added try/catch ctree statement
- decompiler: improved detection of variadic arg types
- decompiler: introduced a new event: `hxe_inlining_func`
- decompiler: published a few graph algorthims (pre/port ordering and dominator calculation)
- decompiler: arm: added support for VSEL instruction (ARMv8-M)
- decompiler: impoved structure copy recognition
- decompiler: improved cfunc_t cache by introducing "saved_to_idb"; otherwise we were saving all decompiled functions upon each "save_database", again and again
- decompiler: improved constant representation in comparisons with binary operators
- decompiler: improved hexrays history to support c++ exception handlers
- decompiler: improved the error message about the missing license: tell the user what license is missing
- decompiler: mips: added support for movtz and movtn (MIPS16e2)
- decompiler: ui: added "Jump to matching brace" action to the context menu
- decompiler: removed welcome form, renamed menu entry to "Hex-Rays Decompiler Options"

Bugfixes

- BUGFIX: ARM: analysis speed could be slow on large 32-bit firmware binaries
- BUGFIX: ARM: comment for UBFIZ instruction was wrong
- BUGFIX: ARM: fixed endless loop which could happen when analysing function chunk before main function entry
- BUGFIX: ARM: fixed CF_JUMP/CALL flags for some instructions (e.g. BLR)
- BUGFIX: ARM: stop decoding undefined MOV Wx, #imm variants (imm not fitting in 32 bits)

- BUGFIX: cvt64: converting an old .idb to .i64 would fail if its path contained a space
- BUGFIX: debugger: win32_remote.exe was unnecessarily requiring an API instroduced in Windows Vista and would not run on XP anymore
- BUGFIX: debugger: win32: IDA's debugger could be detected by a file lock on the modules being loaded into the process
- BUGFIX: debugger: bochs: added support for Bochs 2.8.0
- BUGFIX: decompiler: decompilation of different syscalls in close sequence could be wrong
- BUGFIX: decompiler: expressions with variable sized structures could be mishandled
- BUGFIX: decompiler: IDA could complain "Could not find a matching license for product" when multiple decompilers were installed
- BUGFIX: decompiler: internal errors triggered by UI-related code (e.g. generaing tooltips) could result in "Unknown C++ exception" fatal error
- BUGFIX: decompiler: pressing F5 was not refreshing the pseudocode window in some cases; we were discarding the decompilation result
- BUGFIX: decompiler: value range optimization could lead to code being wrongly removed
- BUGFIX: DSCU: a GAP spanning multiple subcache files would fail to load
- BUGFIX: kernel: IDA on Linux had an unnecessary hard dependency on libsecret and would refuse to run without it.
- BUGFIX: IDA would not mark typical code sequences in raw binary files even if the processor module supported it
- BUGFIX: navigating to a global name which matched a known type name would fail
- BUGFIX: objc: NS*Block reference detection error would end up creating incoherent block structures over unrelated data
- BUGFIX: PC: `alloca_probe` / `chkstk_ms` does not modify rsp or rax in x64 code, unlike x86
- BUGFIX: PC: REX prefix could be incorrectly applied to 32-bit instructions
- BUGFIX: PC: vmovw instruction was decoded as if using 16-bit registers (it actually uses 32-bit ones)
- BUGFIX: PDB: importing types from some large PDBs would fail with "the maximum recursion level was reached"
- BUGFIX: PDB: improved algorithm to extract anonymous(embedded) unions: gap members could be mis-ordered

- BUGFIX: RISCV: fence.i instruction was not decoded
- BUGFIX: SDK: fixed a debug/opt build incompatibility in `reg_finder_t` (due to embedded `std::map` member)
- BUGFIX: SDK: `set_all_bits()` and `clear_all_bits()` would behave wrongly on bitmaps with size not a multiple of 8
- BUGFIX: sometimes information about newly created range-like entities (segments/functions/...) could be lost during UNDO
- BUGFIX: tinfo: xrefs to a deleted enum were not removed
- BUGFIX: UI: default buttons in the 'dark' theme wouldn't stand out
- BUGFIX: UI: editing type of items inside current function was not possible
- BUGFIX: UI: fixed missing scrollbars in the "Output" window when long text was printed
- BUGFIX: UI: large amounts of lines in the "Output" window could cause slowdowns
- BUGFIX: UI: long strings could be truncated when using "Export data"
- BUGFIX: UI: when using `COLOR_INV` color code (e.g. in a custom viewer), IDA would use default color for the text instead of the previous background color
- BUGFIX: UI: quick filters would apply to hidden columns

Last updated 12 days ago