marcobambini / **gravity** Public

Code | Issues 34 | Pull requests 8 | Discussions | Actions | Projects

Commit 18b9195

marcobambini committed 2 days ago · ✓ 5 / 5

```
Bump version to 0.9.6 — OOM safety, init-chain fix, docs, and test suite
improvements

Runtime / VM:
- Add configurable fiber stack size limit (DEFAULT_MAXSTACK_SIZE = 1M slots).
  Infinite recursion and stack exhaustion now produce a clean RUNTIME error
  instead of a hard crash or silent failure.
- Expose GRAVITY_VM_MAXSTACK key via gravity_vm_get/set for runtime tuning.
- Fix gravity_fiber_reassign stack growth: large register-window in $moduleinit
  could push stacktop past the initial allocation (issue #437).

Compiler:
- Fix class $init chain infinite recursion: parent $init helpers ($init2,
  $init3, …) previously used a dynamic name lookup against self, which
  resolved to the wrong override in subclass context. Now emits a direct
  static closure reference (LOADK) via ircode_patch_init_direct.

Tests:
- Re-enable test/disabled/heap.gravity and test/disabled/loop1.gravity,
  moved to test/unittest/. Both now pass with the new OOM error reporting.
- Fix two bugs in heap.gravity (wrong variable assigned, wrong constructor called).
- Add regression test for issue #437 (bugfix_stack_overflow_large_regwin).
- Delete test/disabled/ directory.

Docs:
- Add CHANGELOG.md covering all versions from 0.2.8 to 0.9.6.
- Update README: CMake build instructions, Testing section, Embedding API
  example, CHANGELOG link, replace Gitter with GitHub Discussions.
- Update CLAUDE.md: document fuzzy/infiniteloop test dirs and gravity_vm_get/set.
```

master · 0.9.7 0.9.6

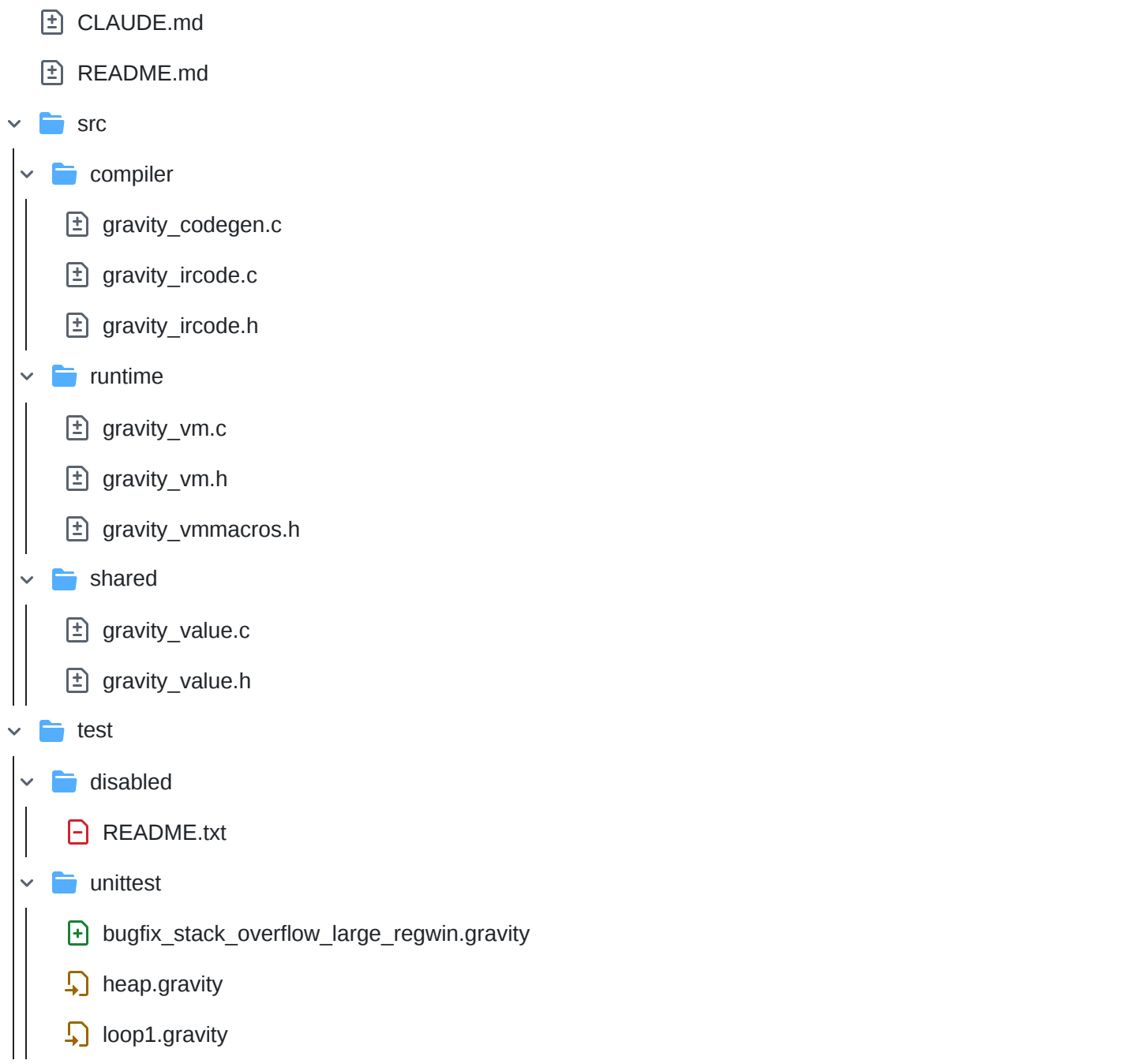
1 parent 87c9c90 commit 18b9195

15 files changed +404 -22 lines changed

↑ Top

Filter files…

CHANGELOG.md



15 files changed +404 -22 lines changed

Search within code

CHANGELOG.md

```
@@ -0,0 +1,194 @@
+ # Changelog
+
+ All notable changes to Gravity are documented in this file.
+
+ ## [0.9.6] - Unreleased
+
```

```
+ ### Fixed
+ - **Stack overflow now produces a clean runtime error** instead of a hard crash (SIGSEGV). Infinite recursion and pathological call depths are caught by a configurable stack size limit before the process runs out of memory.
+ - **Class `$init` chain infinite recursion** — parent-class `$init` helpers (`$init2`, `$init3`, …) previously used a dynamic name lookup against `self`, which resolved to the wrong (overriding) function when called from a subclass, producing infinite recursion. The compiler now emits a direct static closure reference (`LOADK`) so dispatch is always to the correct ancestor function.
+ - **Fiber stack growth in `gravity_fiber_reassign`** — a large register-window allocation in `$moduleinit` could cause the initial stack pointer to overshoot `DEFAULT_MINSTACK_SIZE` (256 slots), leaving `stacktop` pointing into unallocated memory (issue #437).
+ - **`gravity_opt_free` double-free** — optional module cleanup now checks the reference count before freeing (PR #436).
+
+ ### Added
+ - `GRAVITY_VM_MAXSTACK` — runtime-configurable maximum fiber stack size (default 1 048 576 slots / 16 MB). Readable and writable via `gravity_vm_get` / `gravity_vm_set` with key `"maxStack"`.
+ - Re-enabled two previously disabled tests (`heap.gravity`, `loop1.gravity`) — both now pass with the new OOM error reporting.
+
+ ### Changed
+ - Version bumped to **0.9.6** (`GRAVITY_VERSION`, `GRAVITY_VERSION_NUMBER`).
+
+ ---
+
+ ## [0.9.5] - 2024
+
+ ### Fixed
+ - Numerous memory leaks and use-after-free errors throughout the compiler and runtime.
+ - Memory safety improvements across GC, value handling, and object lifecycle.
+ - Clang build compatibility (PR #435).
+
+ ### Changed
+ - Documentation updates: ARCHITECTURE.md rewritten; README refreshed.
+
+ ---
```

```
+
+ ## [0.9.0] - 2023
+
+ ### Fixed
+ - Several memory leaks plugged across the compiler pipeline.
+ - Missing Makefile dependencies (PR #431).
+ - `File.read()` now returns `null` when zero characters are read.
+ - Replaced unsafe `printf` calls with `snprintf`.
+ - Fixed issue #394.
+
+ ### Added
+ - New unit test for leak-related regression coverage.
+
+ ---
+
+ ## [0.8.5] - 2022
+
+ ### Fixed
+ - Setter issue that affected several unit tests.
+ - File read size mismatch due to line-ending differences on Windows (PR #365).
+ - Compilation failure introduced by `O_BINARY` on non-Windows platforms.
+
+ ### Added
+ - Unit tests integrated into CI (PR #378).
+ - BSD shared-object build support; removed `WITH_GETLINE` (PR #375).
+
+ ---
+
+ ## [0.8.4] - 2022
+
+ ### Fixed
+ - Issue #379.
+ - Hash table header organisation (PR #369).
+
+ ---
+
+ ## [0.8.3] - 2021
+
+ ### Fixed
+ - Regression introduced by lazy-loading of superclasses in 0.8.2.
```

```
+
+ ---
+
+ ## [0.8.2] - 2021
+
+ ### Added
+ - Lazy loading of extern superclasses at runtime.
+ - Preliminary support for instance `deinit` (destructor).
+ - `System.input()` (PR #342).
+ - `ENV.argc` / `ENV.argv` properties (PR #343).
+ - ObjC binding example.
+ - C++ binding example.
+ - Ternary expression and `switch` statement codegen (PR #336).
+ - Optional `File` class (cross-platform).
+ - `gravity_instance_lookup_real_property` helper.
+ - `gravity_config.h` for platform-specific configuration (PR #301).
+ - Improved CMake: supports CLI, shared lib, and static lib targets (PR #299).
+
+ ### Fixed
+ - Inner class constructor returning wrong instance.
+ - Computed property (setter) bug.
+ - Sign-conversion and char-type warnings flagged by sanitizers.
+ - `stat` return-value check.
+ - `uint32_t`-to-`char` conversion in `utf8_encode`.
+ - `size_t` comparison against negative value in `file_read`.
+ - Various Windows / MSVC compatibility fixes.
+ - Implicit `long`-to-`double` conversion warning under Clang.
+ - Emscripten include-guard fix.
+
+ ### Changed
+ - Improved superclass type checking in the semantic analyser.
+ - Improved error handling and detection (0.8.0).
+ - `File.eof` renamed from `isEOF`.
+
+ ---
+
+ ## [0.7.9] - 2020
+
+ ### Fixed
+ - Improved error handling in the VM and runtime.
```

```
+
+ ### Added
+ - `vm` back-reference stored on `gravity_closure_t`.
+ - Computed-goto support for Clang on Windows.
+ - `DISPATCH_INNER` macro for `do/while(0)` loops without computed goto.
+ - `xdata` parameter on `delegate->optional_classes` (PR #307).
+
+ ---
+
+ ## [0.7.8] - 2020
+
+ ### Added
+ - Preliminary `Struct` support.
+ - `bind` method fix; unit test added.
+
+ ### Fixed
+ - Possible GC issue (unit test added).
+
+ ---
+
+ ## [0.7.7] - 2020
+
+ ### Fixed
+ - `super` keyword resolution issue; unit test added.
+
+ ---
+
+ ## [0.7.6] - 2020
+
+ ### Changed
+ - Optional classes renamed for consistency.
+
+ ### Fixed
+ - Setter unwanted side effect.
+
+ ---
+
+ ## [0.7.5] - 2020
+
+ ### Improved
```

```
153 + - `float`/`double` to `String` conversion accuracy.
154 + - Various core methods.
155 +
156 + ---
157 +
158 + ## [0.7.4] - 2020
159 +
160 + ### Fixed
161 + - `String.length` is now UTF-8 aware; `String.bytes` added (unit test added).
162 + - Function returning address of local variable on Windows (`directory_read`).
163 + - Division-by-zero warning suppression in GCC.
164 + - Const output-buffer issue on Windows (`WideCharToMultiByte`).
165 +
166 + ### Added
167 + - More BSD targets in `make` and CMake.
168 +
169 + ---
170 +
171 + ## [0.7.0] - 2019
172 +
173 + ### Added
174 + - `String.split` and string iteration are now Unicode-aware; unit test added.
175 + - Support for local `enum` declarations; unit test added.
176 +
177 + ### Fixed
178 + - Local class declarations.
179 + - Superclass resolution edge cases.
180 + - `continue` keyword inside `for` loops.
181 + - `self` parameter in complex postfix expressions.
182 + - Comparison between different object types no longer raises a spurious runtime
      error.
183 +
184 + ---
185 +
186 + ## [0.6.x] - 2018–2019
187 +
188 + Series of incremental releases adding language features (closures, ranges,
      maps, lists, optional modules) and fixing compiler and runtime issues. See git
      history for per-commit details.
189 +
```

```
+ ---
+
+ ## [0.5.x] - 2017–2018
+
+ Initial public release series establishing the core language, VM, and compiler pipeline.
```

CLAUDE.md

```
@@ -29,6 +29,10 @@ Compiler flags: `-std=gnu99 -fgnu89-inline -fPIC -DBUILD_GRAVITY_API`
./gravity -i 'print("hello")'          # Inline execution
```

+ The `test/` directory also contains:
+ - `test/fuzzy/` — randomised fuzzing inputs; all must compile/run without crashing
+ - `test/infiniteloop/` — programs that must terminate with a `RUNTIME` error (not hang)
+
CI runs: `make && test/unittest/run_all.sh`

Architecture
@@ -59,6 +63,7 @@ Core API in `src/runtime/` and example usage in `examples/example.c`:
- `gravity_compiler_create/run` — compile source to closures
- `gravity_vm_new/runmain/runclosure` — create VM and execute code
- `gravity_vm_loadfile/loadbuffer` — load from file or memory
+ - `gravity_vm_get/gravity_vm_set` — read/write VM configuration at runtime (e.g. `GRAVITY_VM_MAXSTACK` to cap fiber stack growth)

Code Style
```

README.md

```
@@ -73,6 +73,7 @@ func main() {

Building
```
```

```
      76  + **Make (Linux / macOS / BSD)**
76    77    ```bash
77    78    make                    # Build the gravity CLI executable
78    79    make mode=debug         # Debug build with symbols
          @@ -81,6 +82,15 @@ make example # Build the C embedding API example
81    82    make clean              # Clean all build artifacts
82    83    ```
83    84
      85  + **CMake (cross-platform, including Windows)**
      86  + ```bash
      87  + cmake -B build
      88  + cmake --build build
      89  + # Optionally disable the CLI and build the library only:
      90  + cmake -B build -DBUILD_CLI=OFF
      91  + cmake --build build
      92  + ```
      93  +
84    94    Requires a C99 compiler. No external dependencies.
85    95
86    96    ## Usage
          @@ -94,6 +104,16 @@ Requires a C99 compiler. No external dependencies.
94   104    ./gravity -t test/unittest              # Run unit tests
95   105    ```
96   106
     107  + ## Testing
     108  +
     109  + ```bash
     110  + ./gravity -t test/unittest              # Run all unit tests via the VM
     111  + ./test/unittest/run_all.sh              # Run all unit tests via shell script
            (with per-test timeouts)
     112  + ./gravity test/unittest/somefile.gravity # Run a single test file
     113  + ```
     114  +
     115  + The `test/` directory also contains `fuzzy/` (randomised fuzzing inputs) and
            `infiniteloop/` (tests that must terminate with a runtime error rather than
            hang).
     116  +
97   117    ## Project Structure
98   118
99   119    ```
```

```
@@ -108,6 +128,48 @@ src/
108 128
109 129  For a comprehensive technical deep-dive into the implementation, see [ARCHITECTURE.md](ARCHITECTURE.md).
110 130
    131 + ## Embedding API
    132 +
    133 + Gravity is designed to be embedded inside a host application. The complete API lives in `src/runtime/gravity_vm.h` and `src/compiler/gravity_compiler.h`. A minimal example:
    134 +
    135 + ```c
    136 + #include "gravity_compiler.h"
    137 + #include "gravity_core.h"
    138 + #include "gravity_vm.h"
    139 +
    140 + static void report_error(gravity_vm *vm, error_type_t type,
    141 +                          const char *description, error_desc_t desc, void *xdata) {
    142 +     printf("%s\n", description);
    143 + }
    144 +
    145 + int main(void) {
    146 +     const char *source = "func main() { return 6 * 7; }";
    147 +
    148 +     gravity_delegate_t delegate = {.error_callback = report_error};
    149 +
    150 +     // compile
    151 +     gravity_compiler_t *compiler = gravity_compiler_create(&delegate);
    152 +     gravity_closure_t *closure   = gravity_compiler_run(compiler, source, strlen(source), 0, true, true);
    153 +
    154 +     // create VM and transfer compiler-owned objects into it
    155 +     gravity_vm *vm = gravity_vm_new(&delegate);
    156 +     gravity_compiler_transfer(compiler, vm);
    157 +     gravity_compiler_free(compiler);
    158 +
    159 +     // execute and read result
    160 +     if (gravity_vm_runmain(vm, closure)) {
    161 +         gravity_value_t result = gravity_vm_result(vm);
```

```
+         gravity_value_dump(vm, result, NULL, 0);  // prints: 42
+     }
+
+     gravity_vm_free(vm);
+     gravity_core_free();
+     return 0;
+ }
+ ```
+
+ See [`examples/example.c`](examples/example.c) and the [embedding documentation](https://gravity-lang.org) for the full bridging API.
+
  ## Special thanks
  Gravity was supported by a couple of open-source projects. The inspiration for closures comes from the elegant <a href="http://www.lua.org" target="_blank">Lua</a> programming language; specifically from the document <a href="http://www.cs.tufts.edu/~nr/cs257/archive/roberto-ierusalimschy/closures-draft.pdf">Closures in Lua</a>. For fibers, upvalues handling and some parts of the garbage collector, my gratitude goes to <a href="http://journal.stuffwithstuff.com" target="_blank">Bob Nystrom</a> and his excellent <a href="https://github.com/munificent/wren">Wren</a> programming language. A very special thanks should also go to my friend **Andrea Donetti** who helped me debugging and testing various aspects of the language.

@@ -118,8 +180,15 @@ The <a href="https://marcobambini.github.io/gravity/#/README">Getting Started</a
  * Gravity is the core language built into Creo (https://creolabs.com)
  * Gravity is the scripting language for the Untold game engine (https://youtu.be/OGrWq8jpK14?t=58)

+ ## Changelog
+
+ See [CHANGELOG.md](CHANGELOG.md) for a summary of changes across versions.
+
  ## Community
- Seems like a good idea to make a group chat for people to discuss Gravity.<br> [![Join the chat at https://gitter.im/gravity-lang/](https://badges.gitter.im/Join%20Chat.svg)](https://gitter.im/gravity-lang/Lobby?utm_source=badge&utm_medium=badge&utm_campaign=pr-badge&utm_content=badge)
```

```
      188  +
      189  + [![GitHub Discussions](https://img.shields.io/badge/discussions-GitHub-blue)]
             (https://github.com/marcobambini/gravity/discussions)
      190  +
      191  + Questions, ideas, and general discussion are welcome in [GitHub Discussions]
             (https://github.com/marcobambini/gravity/discussions).
123   192
124   193    ## Contributing
125   194    Contributions to Gravity are welcomed and encouraged!<br>
```

src/compiler/gravity_codegen.c

```
             @@ -929,11 +929,19 @@ static void process_constructor (gvisitor_t *self,
             gravity_class_t *c, gnode_t *
929   929              char name[256];
930   930              snprintf(name, sizeof(name), "%s%d", CLASS_INTERNAL_INIT_NAME,
                       ninit++);
931   931
932         -            // add new internal init to class and call it from main $init
                       function
933         -            // super_init should not be duplicated here because class hash
                       table values are not freed (only keys are freed)
      932  +            // keep the class binding for serialisation compatibility
934   933              gravity_class_bind(c, name, VALUE_FROM_OBJECT(super_init));
935         -            uint16_t index = gravity_function_cpool_add(NULL,
                       internal_init_function, VALUE_FROM_CSTRING(GET_VM(), name));
936         -            ircode_patch_init((ircode_t *)internal_init_function->bytecode,
                       index);
      934  +
      935  +            // store the closure directly in the constant pool so the call does
                       not
      936  +            // need a runtime name lookup via LOAD from self.  A dynamic lookup
      937  +            // lets parent $initN names resolve against the subclass's hash
                       table,
      938  +            // where they alias to different (deeper) $init functions, causing
      939  +            // infinite recursion when a subclass is instantiated.
      940  +            // gravity_closure_new already registers the closure with GET_VM();
                       pass
      941  +            // NULL to gravity_function_cpool_add to avoid registering it a
                       second time
```

```
942 +             gravity_closure_t *super_closure = gravity_closure_new(GET_VM(), super_init);
943 +             uint16_t index = gravity_function_cpool_add(NULL, internal_init_function, VALUE_FROM_OBJECT(super_closure));
944 +             ircode_patch_init_direct((ircode_t *)internal_init_function->bytecode, index);
937 945           }
938 946           super = super->superclass;
939 947       }
```

Comments 0

Please sign in to comment.