Mono Debugger

Martin Baulig
martin@novell.com

October 25, 2006

Novell.

Summary

Debugging applications in UNIX and .NET

Two debugger models

The Mono Debugger

Threading Model

Breakpoints

Current Status

About Debuggers

Debugging applications in UNIX and .NET

UNIX applications

- Addresses of methods known at compilation time.
- Compiler creates symbol file for the debugger

.NET applications

- Methods compiled by the JIT
- Debugging info must be created dynamically
- Breakpoints in code which isn't JITed yet

Two debugger models

In-Proc

- Debugger and debuggee in same process
- Debugger crashes with debuggee
- Can't debug native code
- Communication with JIT easier

- → **.NET-Debugger**

Out-of-proc

- Debugger in separate process
- Can inspect state from the outside
- Can debug native code
- Need special JIT interface

- → **MDB**

The Mono Debugger

The Mono Debugger

- Written almost entirely in C#
- Doesn't need to talk to an external debugger
- Modular frontend / backend design
  - Architecture / platform specific code in small C library
  - User interface completely platform independ
- Can debug managed and unmanaged code

Threading Model

- Fully multi-threading
- One SSE per thread -> can single-step individual threads
- breakpoints can be inserted per-thread

Breakpoints

- Breakpoints can be inserted in managed and unmanaged code
- Method doesn't need to be JITed
- Per-thread breakpoints

Current Status

Last release was 0.30 "Koblenz".

Command-Line version pretty much stable and usable.

After a longer generics-related break, I'm now back working full-time on the debugger.

Needs more testing and feedback.

Short Demonstration

Future Plans

MonoDevelop integration

· Need to split the debugger into two parts: client and server process

· Can't run the debugger's main loop inside the MonoDevelop process

· Work started now, but will need some time to complete

Other stuff

- XSP integration
- Debug websites

Questions

Ask questions.

Martin Baulig
martin@novell.com