The GNU* Project Debugger - GDB
# Intel® System Studio 2014 for Linux*

Support for Latest Intel Processors & SoCs

<table>
<thead>
<tr>
<th>Software</th>
<th>Silvermont Microarchitecture</th>
<th>Ivy Bridge Microarchitecture</th>
<th>Haswell Microarchitecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® JTAG Debugger† System Debug</td>
<td>✓</td>
<td>--</td>
<td>✓</td>
</tr>
<tr>
<td>Enhanced GDB* Debugger Application Debug</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Intel® Inspector – Memory &amp; Thread Analysis</td>
<td>Memory &amp; Thread Analysis</td>
<td>Memory &amp; Thread Analysis</td>
<td>Memory &amp; Thread Analysis</td>
</tr>
<tr>
<td>Intel® VTune™ Amplifier † † Power &amp; Performance</td>
<td>Hardware Events</td>
<td>Hardware Events</td>
<td>Hardware Events</td>
</tr>
<tr>
<td>Intel® C++ Compiler</td>
<td>SSE4.2</td>
<td>SSE, AVX</td>
<td>SSE, AVX, AVX2, FMA3</td>
</tr>
<tr>
<td>Intel® MKL</td>
<td>--</td>
<td>SSE, AVX</td>
<td>SSE, AVX, AVX2, FMA3</td>
</tr>
<tr>
<td>Intel® IPP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

† Hardware platform debug coverage added as new processors ship
† † Hardware events for new processors added as new processors ship

*Other brands and names are the property of their respective owners.
What you will learn from this slide deck

• GDB technical training
  • Linux* and Tizen™

• In-depth explanation of for Eclipse* development environment with Windows* Host

• Please see subsequent slide decks for in-depth technical training on other components
Windows* Host and Eclipse* Integration
Windows* Host Build and Debug Support with Eclipse* Integration

Windows* Host Eclipse* integration for SVEN, GDB and Intel® C++ Compiler
GUI Installer for Windows* and Linux*
## Updated Studio Components

<table>
<thead>
<tr>
<th>Build</th>
<th>Major Update</th>
<th>Windows* &amp; Linux* Host</th>
<th>Eclipse* Integration</th>
<th>Tizen*</th>
<th>Yocto Project* 1.4</th>
<th>Intel® Atom™ Processor E3xxx</th>
<th>4th Gen Intel® Core™ Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® C++ Compiler <strong>14.0</strong> for Embedded OS Linux*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Intel® Integrated Performance Primitives <strong>8.0 Update 1</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Intel® Math Kernel Library <strong>11.1</strong> (Intel® 64 only)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel® VTune™ Amplifier <strong>2014</strong> for Systems</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Intel® VTune™ Amplifier Sampling Enabling Product (SEP) <strong>3.10 Update 12</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel® Inspector <strong>2014</strong> for Systems</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The GNU* Project Debugger – GDB v7.5 (Provided under GNU Public License v3)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Intel® JTAG Debugger <strong>2014</strong> for Linux*</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel® JTAG Debugger notification module <strong>xdbntf.ko</strong> (Provided under GNU Public License v2)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVEN Technology 1.0 (Provided under GNU Public License v2)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Other brands and names are the property of their respective owners.*
**Supported OSs**

**Host:**
- Red Hat Enterprise* Linux* 5, 6
- Ubuntu* 10.04 LTS, 12.04 LTS, **13.04**
- Fedora* 17, 18
- **Wind River* Linux* 4, 5**
- openSUSE 12.1
- SUSE LINUX Enterprise Server* 11 SP2
- **Microsoft* Windows* 7,8**

**Target:**
- Yocto Project* 1.3, 1.4, and newer based environment
- CE Linux* PR32 based environment
- **Tizen* IVI 1.0, 2.0**
- Wind River* Linux* 4, 5 based environment
Intel® Atom™ Processor supports Branch Trace Store (BTS) using cache-as-RAM or system DRAM

Set breakpoint in OS signal event handler
Unroll execution flow leading up to stack overflow or segmentation fault
Follow execution backwards to where it deviated from expectation
Rerun to that point and analyze memory accesses

Unroll past execution flow:
Where did things start to go wrong?
btrace: h/w supported branch tracing

(gdb) btrace list 1-3
1 in foo () at foo.c:23-28
2 in foo () at foo.c:21-22
3 in bar () at foo.c:18

(gdb) btrace /m 3
foo.c:18: foo (42)
0x400515 <bar+15>: mov $0x2a, %rax
0x400519 <bar+19>: call 0x400400 <foo>

• H/W records per-thread branch trace.
• Show compact control flow overview.
• Show detailed execution trace disassembly.

Answers “how did I get here?”
**btrace: command line syntax**

1. `btrace enable/disable [all] [auto] [<num>[-<num>]]`:

   start/stop recording the branch trace for the current [all] [new] [listed] thread[s]

1. `btrace list [/a] [/f] [/l] [/t] [<num>[-<num>]]`:

   display a list of blocks

1. `btrace [/n] [/m] [+, -, <num>[-<num>]]`:

   display the disassembly of one block

"help branchtracing " for syntax details
btrace: use cases

Debug internal state corruption.

Debug stack corruption (broken backtrace).

Quick control flow overview.
btrace: Example

(gdb) btrace enable auto

program crashed and back trace is not much help:

(gdb) run
Starting program: ../gdb/trace/examples/function_pointer-stack64 Program received signal SIGSEGV, Segmentation fault. 0x0000000000000002a in ?? ()
(gdb) bt
#0 0x0000000000000002a in ?? ()
#1 0x000000000000000017 in ?? ()
#2 0x00000000000040050e in fun_B (arg=0x4005be) at src/stack.c:32
#3 0x0000000000000000 in ?? ()

Look at the branch trace.
List of blocks starts from the most recent block (ending at the current pc) and continues towards older blocks such that control flows from block n+1 to block n.

(gdb) btrace list 1-7
1 in ?? ()
2 in fun_B () at src/stack.c:36-37
3 in fun_B () at src/stack.c:32-34
4 in main () at src/stack.c:57
5 in fun_A () at src/stack.c:22-25
6 in fun_A () at src/stack.c:18-20
7 in main () at src/stack.c:51-56

from main(), we called first fun_A() and then fun_B(). The call to fun_A() returned, and we crashed somewhere in fun_B().
btrace: Example

Look at the disassembly of the last 3 blocks in original control flow (i.e. reverse trace) order, starting from the call to fun_B() from main().

/m interleaves source info

(gdb) btrace /m 1-3
src/stack.c:32  static long fun_B(void* arg) {
  0x0000000000040050 <fun_B+1>:   mov  %rsp,%rbp
  0x0000000000040051 <fun_B+4>:   mov  %rdi,-0x18(%rbp)
src/stack.c:33  struct B_arg* myarg = arg;
  0x0000000000040055 <fun_B+8>:   mov  -0x18(%rbp),%rax
  0x0000000000040059 <fun_B+12>:  mov  %rax,-0x8(%rbp)
src/stack.c:34  if (!myarg) return -1;
  0x000000000004005d <fun_B+16>:  cmpq  $0x0,-0x8(%rbp)
  0x0000000000040061 <fun_B+21>:  jne  0x40052d <fun_B+32>
src/stack.c:36  return myarg->arg1 + myarg->arg2;
  0x0000000000040065 <fun_B+32>:  mov  -0x8(%rbp),%rax
  0x0000000000040069 <fun_B+36>:  mov  (%rax),%rdx
  0x000000000004006d <fun_B+39>:  mov  -0x8(%rbp),%rax
  0x0000000000040071 <fun_B+43>:  mov  0x8(%rax),%rax
  0x0000000000040075 <fun_B+47>:  lea  (%rdx,%rax,1),%rax
src/stack.c:37  }
  0x0000000000040079 <fun_B+51>:  leaveq
  0x000000000004007b <fun_B+52>:  retq
⇒0x000000000000002a: Cannot access memory at address 0x2a

fun_B() is executed and returns to an invalid address suggesting a corrupted stack. fun_B() leaves but that there was no corresponding push on entry to fun_B() => the function pointer comp that was called in main() had been corrupted.
Data Race Detection

Compile with –debug parallel (icc/icpc/ifort only)
Debugger breaks when race has been detected

(gdb) pdbx enable
(gdb) c
data race detected
1: write shared, 4 bytes from foo.c:36
3: read shared, 4 bytes from foo.c:40

Breakpoint -11, 0x401515 in L_test___21 () at foo.c:36
36*var = 42; /* bp.write */

Stop in the context of the racing access
Data Race Detection: Filters

Filters and filter sets to fine-tune the analysis
• Suppress: ignore specified regions
• Focus: ignore non-specified regions

(gdb) pdbx filter line foo.c:36
(gdb) pdbx filter code 0x40518..0x40524
(gdb) pdbx filter var shared
(gdb) pdbx filter data 0x60f48..0x60f50
(gdb) pdbx fset list
focus filter set bug0815, 1 filters
focus filter set bug42, 2 filters
suppress filter set default, 0 filters

Filter all read accesses
(gdb) pdbx filter reads

Lots of useful filter operations
Date Race Detection: Manage Overhead

Race detection is expensive – very expensive
• Both in performance and memory consumption

Manage overhead effectively through
• Selective enabling
• Focus filter sets
• Global filter on reads

It’s an interactive debug tool – use it that way!

Effective filter management is key to fast data race debugging.
Intel® Inspector 2014 for Systems and GDB* work hand-in-hand

• Two-step approach for debugging data races
  - Step 1: whole-program analysis using Intel® Inspector 2014 for Systems
    - Gives a list of all data races in the program
  - Step 2: reproduce individual races in GDB*
    - Select race to debug
    - Start a new focus filter set
    - For each source line in the race report:
      - Add a filter for that line

• GDB* will now break at each instance of the selected data race with very low overhead
GDB* Eclipse Integration

Selecting the Intel provided GDB in Eclipse*
Summary / Next Steps
Intel® System Studio 2014

1. **Windows* Host** and Linux* Host Support with Eclipse* Integration
2. Support for Intel® Atom™ Processor E3xxx (Baytrail SoC)
3. Support for Tizen* IVI
4. Intel® C++ Compiler Improvements
   - Optimizations for **latest Intel processors**
   - **improved sysroot** and GNU cross-build integration support
   - Cross-build support for Windows* host and Wind River* Linux* target
5. Extended Intel® VTune™ Amplifier **System-Wide Analysis**
6. **Yocto Project* Compatible**
7. Intel® JTAG Debugger support for next generation **processors of all sizes**
8. Graphical installer for both Windows* host and Linux* host

---

http://intel.ly/system-studio

The next step in Intelligent Systems Software Development
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY
ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS
DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR
IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES
RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY
PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on
Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using
specific computer systems, components, software, operations and functions. Any change to any of
those factors may cause the results to vary. You should consult other information and performance
tests to assist you in fully evaluating your contemplated purchases, including the performance of that
product when combined with other products.

Copyright © Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Core, VTune, and Cilk
are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that
are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and
other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on
microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended
for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for
Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information
regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Copyright© 2012, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.