MAKING YOUR DEBUGGING EFFORTS COUNT:
BEST PRACTICES WITH THE CDT DEBUGGER

Marc Khouzam
ABOUT ME

➢ Working with CDT Debug since 2007

➢ CDT project co-lead, lead for Debug component

➢ Things you don't like about CDT Debug are probably my fault

➢ You can help get them improved
  ➢ Give feedback
  ➢ Open bugs
  ➢ Contribute
AGENDA

› Running the debugger
› No more *Printf*-debugging
› Examining debugging data
› Controlling execution
› Multi-thread and beyond
› More advanced topics
› Future plans

And one demo chosen by YOU
STAND-ALONE DEBUGGER

› Easy installation through its own package
Easy launch: ./cdtdebug -e myBinary
POST-MORTEM DEBUG

› Examining a core file: Variables, Registers, Memory
› ./cdtdebug -c coreFile -e matchingBinary
PROJECT-LESS DEBUG

Debug any binary!

No need to create a project
No more printf debugging
PRINTF DEBUGGING

- Still much too popular
  - Comfortable, familiar, easy

- Costly efficiency limitations
  - Expensive debug cycle
    1. Recompiling
    2. Redeploying to target
    3. Repeating steps to reproduce issue
  - Info provided is fixed per debug cycle
  ➔ Multiple such debug cycles
Familiarity meets flexibility and efficiency!
DYNAMIC-PRINTF

› Printf dynamically inserted by debugger in executing program

› Prints in same location as compiled-printfs

› Same syntax as compiled-printf

› No recompiling! No redeploying!
DYNAMIC_PRINTF

- Handled as CDT breakpoints
EXAMINING DEBUGGING DATA
› In-hover expression view
› Detail pane
› User can modify data directly
STL classes inspect poorly e.g., Vector, List, Map
PRETTY-PRINTING

› Pretty-printers provided with STL library

› Values of elements can even be modified by user!
PER-ELEMENT FORMAT

› Ability to set format per element

› Variables, Expressions, Registers views
Modifying data during execution:
- Memory view
- Variables view
- Registers view
- Expressions view
- Hover
Return value shown after step-return

When at line 2, pressing step-return will trigger showing the return value.
RETURN VALUE ON STEP-OVER

› Currently return value shown only after step-return

› Plans to show return value after a step-over

› Could be multiple values for a line such as:
  – add ( multiply(6,2), divide(9, 3) );
Enhanced Expressions

Shell-like pattern-matching for variables and registers

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>i ; =eax ; =array[1,6]</td>
<td>Group-pattern</td>
<td>4 unique matches</td>
</tr>
<tr>
<td>i</td>
<td>int</td>
<td>9</td>
</tr>
<tr>
<td>eax</td>
<td>int32_t</td>
<td>-1</td>
</tr>
<tr>
<td>array[1]</td>
<td>int</td>
<td>1</td>
</tr>
<tr>
<td>array[6]</td>
<td>int</td>
<td>6</td>
</tr>
<tr>
<td>=ebx* ; =um?</td>
<td>Group-pattern</td>
<td>5 unique matches</td>
</tr>
<tr>
<td>ebp</td>
<td>void *</td>
<td>0xbfffe8</td>
</tr>
<tr>
<td>ebx</td>
<td>int32_t</td>
<td>-1073747096</td>
</tr>
<tr>
<td>um2</td>
<td>wchar_t</td>
<td>129 L'\201'</td>
</tr>
<tr>
<td>um3</td>
<td>wchar_t</td>
<td>255 L'ý'</td>
</tr>
<tr>
<td>um4</td>
<td>wchar_t</td>
<td>256 L'Å'</td>
</tr>
<tr>
<td>=array[305-308]</td>
<td>Group-pattern</td>
<td>4 unique matches</td>
</tr>
<tr>
<td>array[305]</td>
<td>int</td>
<td>305</td>
</tr>
<tr>
<td>array[306]</td>
<td>int</td>
<td>306</td>
</tr>
<tr>
<td>array[307]</td>
<td>int</td>
<td>307</td>
</tr>
<tr>
<td>array[308]</td>
<td>int</td>
<td>308</td>
</tr>
<tr>
<td>*</td>
<td>Group-pattern</td>
<td>54 unique matches</td>
</tr>
<tr>
<td>a</td>
<td>int [2]</td>
<td>0xbfffe08</td>
</tr>
<tr>
<td>aba</td>
<td>int</td>
<td>882090000</td>
</tr>
</tbody>
</table>
ENHANCED-EXPRESSIONS

› Support for pattern-matching and expressions groups
› Provides alphabetical sorting

– Pattern-matched local variables
  • =v?r – Show all local vars matching pattern
  • =* – Show all local vars alphabetically

– Array ranges
  • =myarray[30-40] – Show elements 30 to 40
  • =myarray[1-3,20,23-24] – Show elements 1,2,3,20,23,24
Support for defining expressions and expressions groups

- Pattern-matched registers
  - =$xmm* – Show all registers starting with xmm
  - =$* – Show all registers

- Semi-colon-separated groups
  - var1; var2 – Group which children are var1 and var2
  - var1;=* – Show all local vars with var1 being shown first
ENHANCED-EXPRESSIONS

› Super-set of Variables and of Registers views
CONTROLLING EXECUTION
RUN-TO-LINE

› Run-to-line
- Ctrl+R – Execute program until selected code line
- Or right-click on selected line in editor for menu option
STEP-INTO-SELECTION

› Ability to specify *which* method to step into
  – One step to step into 'substract' instead of 5

```c
int result;
int a = 4, b = 5;

add(4,7);
multiply(5,4);

result = subtract( multiply( add( a, b ), 3 ), 5 );
```
Move-to-line: set execution line to selected one

Resume-at-line: move-to-line and automatically resume

From Run menu or editor right-click menu
REVERSE DEBUGGING

STEP PROGRAM BACKWARDS

› Recording of program execution
› Replay in reverse
› Allows to examine past execution without restarting it
› Reverse-step, reverse-resume
› Can use breakpoints set in the 'past'

New buttons to control reverse execution
SOFTWARE RECORDING

› Software recording
  – Code path
  – Variables changes
  – Register changes
  – Memory changes
Reverse Debugging

- Hardware recording
  - Code path only
  - Requires Intel(R) processor
MULTI-THREAD AND BEYOND
NON-STOP DEBUGGING

› Program continues execution while suspending some threads

› Reduced intrusiveness
MULTI-PROCESS DEBUGGING

One gdb controlling many processes

Debug process interactions
Crowded display when program has many threads

- What is really of interest?
- Threads actively being debugged, i.e., suspended
- Enable from preferences
MORE ADVANCED TOPICS
GDB AND GDB CONSOLE

› GDB is the brains behind CDT Debug
› Can use gdb command-line from eclipse
› Currently very basic.
Targeting CDT 9.1 and GDB 7.12 (by September 2016)
Disassembly View

- Shows disassembly of code (optionally with source)
- Supports breakpoints like in editor (and dynamic-printf!)
- Step/Resume/Suspend from Disassembly view
FUTURE PLANS
GLOBAL BREAKPOINTS

› Contribution to Linux Kernel ongoing

- Applies to every process
- Auto attach when hit
- Un-started or short lived process
• IT Sets to control groups of elements
• About multicore
  – Step group of threads or processes
  – Set breakpoint on a subset of threads
  – Resume execution on a core or set of cores
MORE IDEAS

- Improved handling of breakpoints
  - Showing each installed location per breakpoint
  - ...

- Improved Memory view
  - Showing registers and variables
  - ...

- Evolving Visualizer
  - Better support when dealing with hundreds of cores
  - ...
CONCLUSION
CONCLUSION

› Don't accept `printf-debugging`. This is 2016!

› Debugger will save you time

› Debugging does not have to be difficult

› Help your team improve
  – Lead by example
  – Share knowledge, success stories
Evaluate the Sessions

Sign in and vote at eclipsecon.org
SOME REFERENCES

› CDT Project, http://www.eclipse.org/cdt


› CDT Debug workgroup
  http://wiki.eclipse.org/CDT/MultiCoreDebugWorkingGroup

› CDT Wiki, http://wiki.eclipse.org/CDT
FINAL Q&A