LLDB in Fedora

Jan Kratochvíl <jan.kratochvil@redhat.com>

2019-01-26

Motivation for LLDB

- Reusing clang++ for C++ parsing
- Performance – DIE vs. CU expansion
- C++ liblldb.so API

Reusing clang++ for C++ parsing

Reusing clang++ for C++ parsing

LLDB using clang++ parser and internal LLDB LLVM-IR evaluation:

```
(lldb) print 1+2
(int) $0 = 3
        (lldb) log enable lldb expr
```

Internal GDB parser + evaluation:

```
(gdb) print 1+2
$1 = 3
```

External GCC compiler:

```
(gdb) compile print 1+2
$2 = 3
        (gdb) set debug compile on
```

Reusing clang++ for C++ parsing

(lldb) expression ...

```
#include <map>
#include <vector>
struct Cache {
  const int i;
  std::vector<int> index;
  Cache(int i_):i(i_) { while (index.size()<10000) index.push_back(rand()); }
  bool operator < (const Cache &other) const { return i<other.i; }
};
int main() {
  std::map<Cache,int> cache;
  for (int i=0;i<10;++i)
    cache[Cache(i)]=i;
  return 0;
}
```

Reusing clang++ for C++ parsing

(lldb) expression ...

```
(gdb) p cache
$1 = std::map with 10 elements = {[{i = 0, index = std::vector of length 10000, capacity
16384 = {1804289383, 846930886, 1681692777, 1714636915, 1957747793,
424238335, 719885386, 1649760492, 596516649, 1189641421, 1025202362,
1350490027, 783368690, 1102520059, 2044897763, 1967513926, 1365180540,
1540383426, 304089172, 1303455736, 35005211, 521595368, 294702567,
1726956429, 336465782, 861021530, 278722862, 233665123, 2145174067,
468703135, 1101513929, 1801979802, 1315634022, 635723058, 1369133069,
1125898167, 1059961393, 2089018456, 628175011, 1656478042, 1131176229,
1653377373, 859484421, 1914544919, 608413784, 756898537, 1734575198,
1973594324, 149798315, 2038664370, 1129566413, 184803526, 412776091,
1424268980, 1911759956, 749241873, 137806862, 42999170, 982906996,
135497281, 511702305, 2084420925, 1937477084, 1827336327, 572660336,
1159126505, 805750846, 1632621729, 1100661313, 1433925857, 1141616124,
```

Reusing clang++ for C++ parsing

```
(lldb) expression …
```

```
(gdb) set print elements 8
(gdb) p cache
$1 = std::map with 10 elements = {[{i = 0, index = std::vector of length 10000, capacity
16384 = {1804289383, 846930886, 1681692777, 1714636915, 1957747793,
424238335, 719885386, 1649760492...}}] = 0, [{i = 1, index = std::vector of length
10000, capacity 16384 = {89057537, 1840048410, 427773756, 762677667,
742585312, 1447032062, 1904054136, 1665967229...}}] = 1, [{i = 2, index =
std::vector of length 10000, capacity 16384 = {851227066, 1907169184, 1672626973,
15224425, 1327707705, 1030165428, 454990854, 1575562599...}}] = 2, [{i = 3,
index = std::vector of length 10000, capacity 16384 = {218746804, 231106831,
1109549898, 2097647419, 2119090367, 1322638762, 947642132, 1611526430...}}] =
3...}
(gdb) _
```

Reusing clang++ for C++ parsing

(lldb) expression ...

- Python pretty printer

```
(gdb) python
>class CachePrinter:
>       def __init__ (self, val):
>               self.val = val
>       def to_string (self):
>               return str (self.val['i'])
>def pretty_printer_lookup (val):
>       if str(val.type.unqualified()) == "Cache":
>               return CachePrinter(val)
>       return None
>gdb.pretty_printers.append(pretty_printer_lookup)
>end
(gdb) print cache
$2 = std::map with 10 elements = {[0] = 0, [1] = 10, [2] = 20, [3] = 30, [4] = 40,
[5] = 50, [6] = 60, [7] = 70, [8] = 80, [9] = 90}
```

Reusing clang++ for C++ parsing

(lldb) expression ...

```
(gdb) p cache[Cache(0)]
A syntax error in expression, near `(0)]'.

(gdb) p cache.cbegin()
Cannot evaluate function -- may be inlined
```

... add dummy calls of iterators and:

```
(gdb) set $it=cache.cbegin()
(gdb) print $it->first.i
Attempt to take address of value not located in memory.
```

Reusing clang++ for C++ parsing

(lldb) expression ...

Add to source: **(++cache.cbegin())->first;**

```
(gdb) printf "%d=%d\n",cache.cbegin()->first.i,cache.cbegin()->second
0=0
(gdb) printf "%d=%d\n",(++cache.cbegin())->first.i,(++cache.cbegin())->second
1=1
(gdb) printf "%d=%d\n",(++++cache.cbegin())->first.i,(++++cache.cbegin())->second
2=2
(gdb) printf "%d=%d\n",(++++++cache.cbegin())->first.i,(++++++cache.cbegin())->second
3=3
```

... not too nice, what else?

- JIT-like C++ command

Reusing clang++ for C++ parsing

(lldb) expression ...

- JIT-like C++ command
- Add to source: **for (const auto &it:cache) (void)it;**

- (lldb) expr for (const auto &it:cache) printf("%d=%d, ",it.first.i,it.second); (void)puts("");
  0=0, 1=10, 2=20, 3=30, 4=40, 5=50, 6=60, 7=70, 8=80, 9=90,

- (gdb) compile code for (const auto &it:cache) printf("%d=%d, ",it.first.i,it.second); puts("");
  gdb command line:1:21: error: specialization of ‘template<class _Tp> class new_allocator’ in different namespace [-fpermissive]
  gdb command line:1:21: error: from definition of ‘template<class _Tp> class new_allocator’ [-fpermissive]
  gdb command line:1:21: error: unhandled TYPE_CODE_ERROR

  or: **No compiler support for language c++.**
  or: lock-up

Performance+ – DIE vs. CU expansion

- DIE = Debug Information Entry

```
readelf -wi:
 <1><53b>: Abbrev Number: 18 (DW_TAG_variable)
   <53c>   DW_AT_name       : stdout
   <540>   DW_AT_decl_file  : 10
   <541>   DW_AT_decl_line  : 138
   <543>   DW_AT_type       : <0x530>
```

- CU = Compilation Unit = one.o = .cpp with all its .h files

Performance+ — DIE vs. CU expansion

- 1.3GB libclang.so.9 built by clang++
  CU size in bytes: avg= 527888 median=479207
- 1.9GB libclang.so.9 built by g++
  CU size in bytes: avg=1004744 median=934319

- GDB: read-in of CU and related CUs (~50)
    - seconds and GBs for one command; and even minutes and tens of GBs
    - difficult to change in the GDB codebase
- LLDB: read-in only of DIE and its tree of DIEs
    - immediate

Performance- — indexes

- Apple OSX has `.apple_names` & co.
- DWARF-4 + `.gdb_index`: Fedoras + RHEL-7
- DWARF-5 with `.debug_names`

- GDB: DWARF-4, **`.gdb_index`**,DWARF-5, `.debug_names`
  - global `.gdb_index` by: `/usr/bin/gdb-add-index <executable>`
  - post-processing, re-reading the DWARF from <executable>

  ```
  53.931s GDB startup without .gdb_index
  70.576s gdb-add-index .gdb_index
   4.315s GDB startup with    .gdb_index
  ```

- LLDB: DWARF-4, DWARF-5, **`.debug_names`**
  - per-CU `.debug_names` by: `clang -gdwarf-5 -mllvm -accel-tables=Dwarf`
  - produced from IR of each CU by: `clang -cc1`
  - missing per-CU .debug_names merging by lld (gold)
- `13.612s` LLDB startup without .debug_names (16 cores)

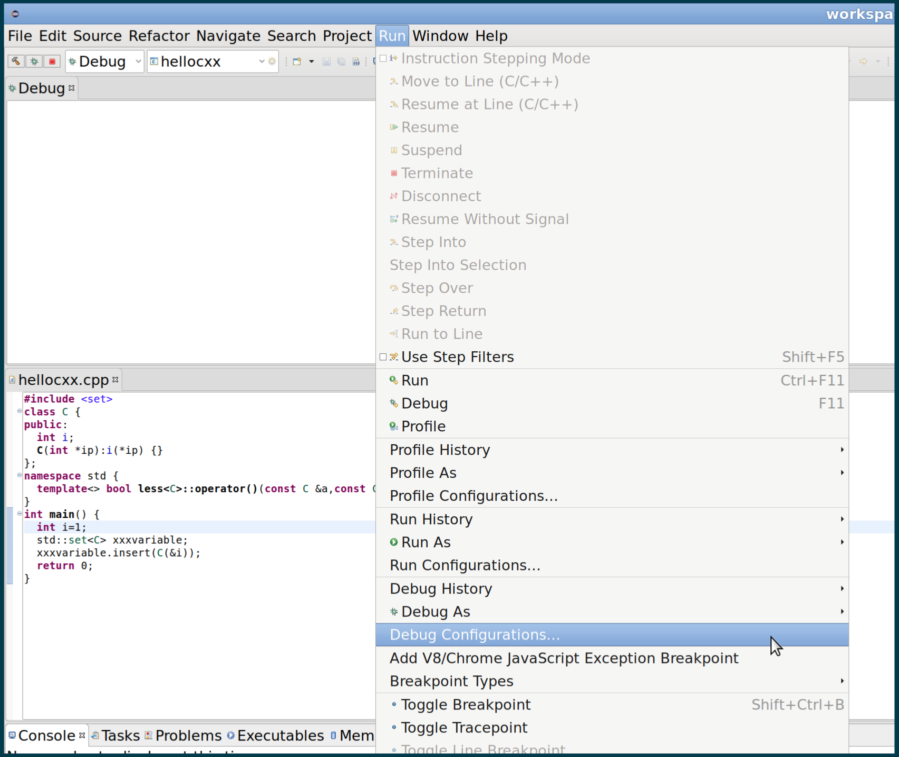
C++ liblldb.so API

no Python!

```
int main(int argc, char **argv) {
  SBDebugger::Initialize();
  ::pid_t pid(atoi(argv[1]));
  SBTarget sbtarget(SBDebugger::Create().CreateTarget(nullptr));
  SBAttachInfo sbattachinfo(pid);
  SBError sberror;
  SBProcess process(sbtarget.Attach(sbattachinfo, sberror));
  for (uint32_t modix=0;modix<sbtarget.GetNumModules();++modix) {
    SBModule sbmodule(sbtarget.GetModuleAtIndex(modix));
    SBValue sbvalue(sbmodule.FindFirstGlobalVariable(sbtarget,
         "main_arena"));
    if (sbvalue.IsValid())
      printf("%#" PRIx64 "\n", sbvalue.GetLoadAddress());
  }
}
```

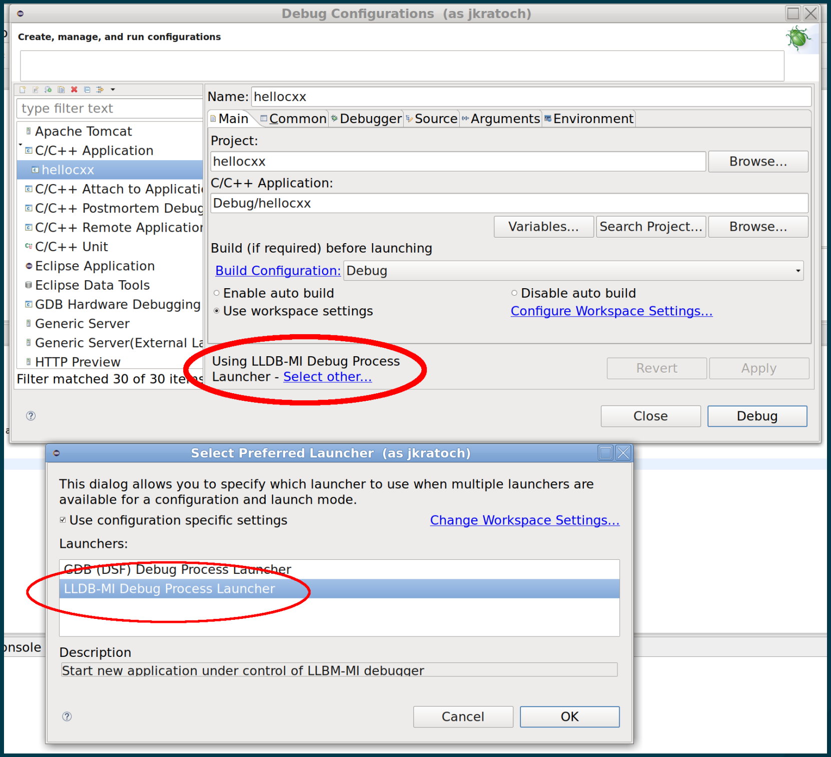
Eclipse CDT integration

`eclipse-cdt-llvm.rpm` must be installed



hellocxx.cpp

```
#include <set>
class C {
public:
  int i;
  C(int *ip):i(*ip) {}
};
namespace std {
  template<> bool less<C>::operator()(const C &a,const C
}
int main() {
  int i=1;
  std::set<C> xxxvariable;
  xxxvariable.insert(C(&i));
  return 0;
}
```



Debug Configurations (as jkratoch)
Create, manage, and run configurations
type filter text
Apache Tomcat
C/C++ Application
hellocxx
C/C++ Attach to Applicati
C/C++ Postmortem Debug
C/C++ Remote Application
C/C++ Unit
Eclipse Application
Eclipse Data Tools
GDB Hardware Debugging
Generic Server
Generic Server(External La
HTTP Preview
Filter matched 30 of 30 items
Main Common Debugger Source Arguments Environment
Browse...
Variables... Search Project... Browse...
Revert Apply
Close Debug

Name: hellocxx

Project: hellocxx

C/C++ Application: Debug/hellocxx

Build (if required) before launching

Build Configuration: Debug

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Using LLDB-MI Debug Process Launcher - Select other...

Select Preferred Launcher (as jkratoch)

This dialog allows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.

☑ Use configuration specific settings Change Workspace Settings...

Launchers:

GDB (DSF) Debug Process Launcher

LLDB-MI Debug Process Launcher

Description

Start new application under control of LLBM-MI debugger

```
15562 pts/2    S+     0:00          \_ eclipse
15580 pts/2    Sl+    1:40              \_ /usr/bin/java -Dosgi.requiredJavaVersion=1.8 -Dosgi.dataAreaRequiresExpli
16476 pts/2    Sl     0:00                  \_ lldb-mi --interpreter mi2 --nx
16538 pts/2    S      0:00                      \_ /usr/bin/lldb-server gdbserver --fd=5 --native-regs --setsid
16575 pts/2    t      0:00                          \_ /home/jkratoch/workspace/hellocxx/Debug/hellocxx
```

Functionality Blockers

Before LLDB is usable on Fedora:

- pending: DWZ support - https://fedoraproject.org/wiki/Features/DwarfCompressor
  - otherwise even crashes inside system libraries are not available
    ```
    warning: (x86_64) /usr/bin/bc unsupported DW_FORM values: 0x1f20 0x1f21
    ```
  - my trunk snapshot rogue build with DWZ support:
    - https://copr.fedorainfracloud.org/coprs/jankratochvil/lldb/package/lldb-experimental/
    - dnf copr enable jankratochvil/lldb;dnf install lldb-experimental;lldb-experimental ...
- pending: `.debug_types` by `-fdebug-types-section` (=kind of DWZ)
  - although not required for rpm-built packages
- some LLDB compatibility fixes with GCC DWARF – just use clang:

```
(lldb) p this
error: warning: '__cplusplus' macro redefined
previous definition is here
error: expected unqualified-id
```

Nice To Have

- GDB CLI + command-line args emulation
- setup LLDB buildbot for Fedora and RHEL
- `.gnu_debugdata` ("minidebuginfo")
- unify data formatters / pretty printers for both GDB and LLDB
  - LLDB ships bundled pretty printers for GNU libstdc++ and LLVM libc++:

```
libstdc++.rpm:
    /usr/share/gcc-8/python/libstdcxx/v6/printers.py
python2-lldb.rpm:
    /usr/lib64/python2.7/site-packages/lldb/formatters/cpp/gnu_libstdcpp.py
    /usr/lib64/python2.7/site-packages/lldb/formatters/cpp/libcxx.py
```

Data Formatters / Pretty Printers

```
  std::vector<int> vec{1,2,3};
(lldb) print vec
(std::vector<int, std::allocator<int> >) $1 = size=3 {
  [0] = 1
  [1] = 2
  [2] = 3
}
(lldb) type category disable cplusplus
(lldb) print vec
(std::vector<int, std::allocator<int> >) $2 = {
  std::_Vector_base<int, std::allocator<int> > = {
    _M_impl = {
      _M_start = 0x0000000000416e70
      _M_finish = 0x0000000000416e7c
      _M_end_of_storage = 0x0000000000416e7c
    }
  }
}
(lldb) _
```

redhat.

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos