IBM Education Assistance for z/OS V2R1

Item: Debug Optimized Code
Element/Component: z/OS UNIX System Services DBX

Material is current as of June 2013

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Migration & Coexistence Considerations
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks

Presentation Objectives

- Optimized debug
  - Breakpoints in Inlined code
    - Stop at line
    - Stop in routine
  - Local Variables Support
  - Source mapping
  - Limitations
- C++ Template support
- Code in includes / C++ Member functions
- List with executable lines highlighted
- Limited scope support

Overview

- This line item allows users to debug routines that have been inlined into the debuggable program. Combined with the implicate program snapshot support this provides a solution to debugging optimized code.
- Problem Statement / Need Addressed
  - The debuggable version of a product is much slower and larger than production code.
- Solution
  - This is the second phase of a solution to allow the users to debug optimized code.
- Benefit/Value
  - DBX can now debug production level code.

Usage & Invocation - Inline Support

```
volatile int x,l,y;
inline int foo(void)
{
   x = 2;
   return(x);
}
int main()
{
   volatile int ti;
   l = 0;
   ti = 0;
   ti = foo();
   ti = 0;
   for( ti = 0; ti < 2; ti++)
            x = foo();
}
```

```
c99 -O2 -g9 example.c

Or

c99 -O2 -qdebug=nohook -g9 example.c
```

Usage & Invocation - Stop in Routine

```
(dbx64) stop in foo
[1] stop in 'int foo()'
(dbx64) cont
[1] stopped in 'int foo()' at line 4 in file "a.c"  ($t1)
    4       x = 2;
(dbx64) list
    5       return(x);
    6   }
    7   int main()
    8   {
    9       volatile int ti;
   10       l = 0;
   11       ti = 0;
   12       ti = foo();
   13       ti = 0;
   14       for( ti = 0; ti < 2; ti++)
(dbx64) cont
[1] stopped in 'int foo()' at line 5 in file "a.c"  ($t1)
    5       return(x);
(dbx64) cont
[1] stopped in 'int foo()' at line 5 in file "a.c"  ($t1)
    5       return(x);
(dbx64) c
FDBX0114: program exited
```

Usage & Invocation - Stop at line

```
(dbx64) list 1,7
    1   volatile int x,l,y;
    2   inline int foo(void)
    3   {
    4      x = 2;
    5      return(x);
    6   }
    7   int main()
(dbx64) stop at 4
[1] stop at "a.c":4
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "a.c"  ($t1)
    4      x = 2;
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "a.c"  ($t1)
    4      x = 2;
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "a.c"  ($t1)
    4      x = 2;
(dbx64) c
FDBX0114: program exited
```

Usage & Invocation - Local Variable Support

```
int x = 2, y = 3;
inline int foo(int aaa, int bbb)
{
    int local_v = 1;
    long vvvv = 100;
    aaa = aaa * bbb;
    local_v = aaa + bbb + x;
    bbb = bbb + 7;
    return aaa / bbb + local_v;
}

int main()
{
    x = foo (x, y);
    x = x * y;
    x = foo (x, x * y);
    y = x + y;
    x = foo (y - x, y + x);
    return x + y;
}
```

```
c99 -O2 -g9 example.c
```

Usage & Invocation - Local Variable Support

```
(dbx64) stop in foo
[1] stop in 'int foo()'
(dbx64) cont
[1] stopped in 'int foo()' at line 4 in file "a.c"  ($t1)
    4       int local_v = 1;
(dbx64) list
    5       long vvvv = 100;
    6       aaa = aaa * bbb;
    7       local_v = aaa + bbb + x;
    8       bbb = bbb + 7;
    9       return aaa / bbb + local_v;
   10   }
   11
   12   int main()
   13   {
   14       x = foo (x, y);
(dbx64) stop at 9
[2] stop at "a.c":9
(dbx64) cont
[2] stopped in 'int foo()' at line 9 in file "a.c"  ($t1)
    9       return aaa / bbb + local_v;
(dbx64) p local_v
11
(dbx64) p vvvv
100
```

Usage & Invocation - Templates

```
template <class T>
T GetMax (T a, T b)
{
    T result;
    result = (a>b)? a-1 : b-1;
    return (result);
}

int main ()
{
    int i=10, j=60, k;
    long l=70, m=50, n;
    k=GetMax<int>(i,j);
    n=GetMax<long>(l,m);
    return 0;
}
```

```
xlC -qdebug=nohook template.C
```

Usage & Invocation - Templates

```
(dbx64) list 1,8
    1   template <class T>
    2   T GetMax (T a, T b)
    3   {
    4       T result;
    5       result = (a>b)? a-1 : b-1;
    6       return (result);
    7   }
    8
(dbx64) stop at 5
[1] stop at "st_15.C":5
(dbx64) c
[1] stopped in 'int GetMax<int>(int a, int b)' at line 5 in file "template.C"  ($t1)
    5       result = (a>b)? a-1 : b-1;
(dbx64) c
[1] stopped in 'long GetMax<long>(long a, long b)' at line 5 in file "template.C" ($t1)
    5       result = (a>b)? a-1 : b-1;
(dbx64) c
FDBX0114: program exited
(dbx64)
```

Usage & Invocation - Member functions

```
(dbx64) file classx.h
(dbx64) list
    1   class x
    2   {
    3      public:
    4        int a,b;
    5        int add()
    6        {
    7            int k;
    8            k = a+b;
    9            return k;
   10        }
(dbx64) stop at 8
[1] stop at "classx.h":8
(dbx64) c
[1] stopped in 'int add()' at line 8 in file "classx.h"  ($t1)
    8            k = a+b;
(dbx64) delete 1
(dbx64) stop in x::add
[2] stop in 'int x::add()'      File /u/dbxteam/classx.h, Line 8
(dbx64) c
[2] stopped in 'int add()' at line 8 in file "classx.h"  ($t1)
    8            k = a+b;
```

classx.h may be included into 100's of *.C files.

Usage & Invocation - List with executable lines highlighted

```
(dbx64) set $showcodelines

(dbx64) list 1,12
     1    class x
     2    {
     3       public:
     4         int a,b;
     5         int add()
     6         {
     7             int k;
*    8             k = a+b;
*    9             return k;
*   10         }
    11    };
    12
(dbxs64)
```

Each line with an * in front of it represents location that a breakpoint may be set.

Using implicit program snapshots at high levels of optimization and low debuggality, breakpoints may only be set on certain executable lines and these lines will be identified with an '*' .

Usage & Invocation - Limited Scope Support

```
(dbx64) list 1,12
     1   int i = 5;
     2
     3   int main()
     4   {
     5      printf("%d\n", i );
     6
     7      int i;
     8      i = 2;
     9      printf("%d\n", i );
    10
    11   }
(dbxs64) step
stopped in main at line 5 in file "1.c"  ($t1)
    5       printf("%d\n", i );
(dbx64) p i
5
(dbx64) step 2
5
stopped in main at line 9 in file "1.c"  ($t1)
    9       printf("%d\n", i );
(dbx64) p i
2
(dbx64)
```

Migration & Coexistence Considerations

- The “stop in <routine>” command will now stop at the first statement in the routine not the first line.
- Code must be recompiled on V2R1 to get this support.

Presentation Summary

- DBX can debug production level code now. This is code compiled with optimization. Support included:
  - Debug of inlined routines
  - Support for implicit program snapshots
  - Template debug
  - Debug of code in include files included into many compile units

Appendix

- SA38-2280 zOS UNIX System Services Command Reference
- SA38-2284 zOS UNIX System Services Messages and Codes
- SA38-2282 zOS UNIX System Services Programming Tools
- SA38-2279 zOS UNIX System Services User's Guide