backtrace command

Displays the call stack for the currently selected thread.

Syntax

```
backtrace
backtrace [Frame count]
backtrace full [Frame count]
bt
bt [Frame count]
bt full [Frame count]
```

Parameters

Frame count

Specifies the amount of frames to display. If this number is positive, GDB will display the specified amount of innermost frames. If it is negative, GDB will display the specified amount of outermost frames. If no frame count is specified, GDB will display the default amount of frames that is configured using the **set backtrace limit** command.

full

If specified, the **backtrace** command will show information about the locals for each frame.

Remarks

The call stack (also known as backtrace) is the information about the current function, the function that called it, the function that called the calling function and so on. Refer to the example below for more details.

Examples

We will show how the **backtrace** command works using the following example:

```
#include <stdio.h>

void level0()
{
    printf("Reached level 0\n");
}

void test(int level)
{
    if (level > 0)
    {
        int prevLevel = level - 1;
        printf("Level %d\n", level);
        test(prevLevel);
    }
    else
        level0();
}

int main()
{
    test(5);
    return 0;
}
```

In this example, main() calls test(5), test(5) calls test(4) and so on until test(0) calls level0(). We will put a breakpoint at level0() and display the backtrace that will show the function chain from the innermost level0() up until the outermost main():

```
(gdb) b level0
Breakpoint 1 at 0x804841a: file recursion.cpp, line 5.
(gdb) r
Starting program: /home/testuser/recursionDemo
Level 5
Level 4
Level 3
Level 2
Level 1

Breakpoint 1, level0 () at recursion.cpp:5
5 printf("Reached level 0\n");
(gdb) backtrace
#0 level0 () at recursion.cpp:5
#1 0x08048462 in test (level=0) at recursion.cpp:17
#2 0x0804845b in test (level=1) at recursion.cpp:14
#3 0x0804845b in test (level=2) at recursion.cpp:14
#4 0x0804845b in test (level=3) at recursion.cpp:14
#5 0x0804845b in test (level=4) at recursion.cpp:14
#6 0x0804845b in test (level=5) at recursion.cpp:14
#7 0x08048479 in main () at recursion.cpp:22
(gdb) backtrace full
#0 level0 () at recursion.cpp:5
No locals.
#1 0x08048462 in test (level=0) at recursion.cpp:17
No locals.
#2 0x0804845b in test (level=1) at recursion.cpp:14
prevLevel = 0
#3 0x0804845b in test (level=2) at recursion.cpp:14
prevLevel = 1
#4 0x0804845b in test (level=3) at recursion.cpp:14
prevLevel = 2
#5 0x0804845b in test (level=4) at recursion.cpp:14
prevLevel = 3
#6 0x0804845b in test (level=5) at recursion.cpp:14
prevLevel = 4
#7 0x08048479 in main () at recursion.cpp:22
No locals.
(gdb) backtrace 2
#0 level0 () at recursion.cpp:5
#1 0x08048462 in test (level=0) at recursion.cpp:17
(More stack frames follow...)
(gdb) backtrace -2
#6 0x0804845b in test (level=5) at recursion.cpp:14
#7 0x08048479 in main () at recursion.cpp:22
```

Compatibility with VisualGDB

You can execute the **backtrace** command using the GDB Session window in Visual Studio. Alternatively you can use the Call Stack window that shows the same information in the user-friendly way.

See also