A Short Primer on Debugger Internals

When using a debugger, the user typically wants to be able to stop the program at various times during execution, print the value of a program symbol, follow the program flow of control by stepping through the source lines in a function, or examine the program execution stack to determine how execution ended up at a particular location.

To enable the user to perform many of these operations, a debugger must have access to a lot of information, much of it related to program blocks, symbols, data types, and source lines. Many debuggers read this information from the object code file and store it in an internal symbol table for easy lookup and traversal.

A debugger uses the information in the symbol table to translate a symbolic location specified by the user into a virtual address. Once a debugger has a virtual address for a program location, it can set a program monitor (such as a breakpoint, trace, or watchpoint) at that location. Once the virtual address for a program symbol is known, a debugger can find its value and display the value according to the symbol's type.

To determine the type and value of a language expression specified by the user, which may range from a simple variable reference to a complex expression, a debugger needs some way to determine if the expression is correct. One way to implement this functionality is with a parser that translates the expression into some sort of abstract syntax or intermediate language tree made up of operator nodes and operand leaves. This tree is then evaluated to determine the type and result value of the expression.

The contents of the symbol table are important to a debugger, but it also needs access to the address space of the target program. On many UNIX systems, debuggers control the target program with the ptrace() system call. With ptrace() a debugger can read from and write to the program's address space, execute the program for one or many instructions, and send a signal to the program. For example, once a debugger has translated a program location into a virtual address, it uses ptrace() to write a special instruction into the instruction stream of the target program. Execution of the special instruction causes the target program to stop and the debugger to regain control of the target program.

†