

# Introduction to gdb

## Learning to use the gnu debugger

*by Derek Schuurman*

Debugging is frequently part of the process of programming. There are several techniques to debug code. Sometimes debugging may be accomplished by sprinkling `printf` statements throughout your code to display the state of your program and variables as it executes. Another way to debug code is to use a special debugging tool that allows you to see what is going on inside your program while it executes. The **gcc** compiler has a powerful debugger called **gdb** (the GNU debugger).

The purpose of this tutorial is to introduce the use of the `gdb`, the gnu debugger. Most integrated development environments (IDEs) provide a friendly interface for using the debugger. In this tutorial we will look at using the debugger from the command line.

1. Open a **terminal** window and create a sample program as shown below to perform simple calculations based on user input:

```
/* Practice assignment
   Name: your name
   This program multiplies 3 numbers provided by the
   user and prints the result to the display */
#include <stdio.h>

int main(void)
{
    float x,y,z;

    printf("\nWhat is the first number? ");
    scanf("%f",&x);
    printf("\nWhat is the second number? ");
    scanf("%f",&y);
    printf("\nWhat is the third number? ");
    scanf("%f",&z);
    printf("\n%f * %f * %f is equal to %f\n", x, y, z, (x*y*z) );
    return 0;
}
```

Add some comments and compile and run the new program.

Practice using the debugger with the code you wrote in the previous step. Load the source file and then re-compile it using the following command:

```
gcc -Wall -g multiply.c -o multiply
```

Note that the extra **-g** option tells the `gcc` compiler to include information in the output file for use by the debugger.

To start the GNU debugger, type:

**`gdb multiply`**

At the prompt you may now enter various commands. To start, set a breakpoint for the start of the `main()` function by typing:

**`break main`**

A breakpoint stops program execution allowing you trace your program step-by-step. To begin running your program type:

**`run`**

The program will halt at the first line in your program which will be high-lighted in the source file. To display the contents of the `x`, `y` and `z` variables type:

**`display x`**  
**`display y`**  
**`display z`**

Note that before these variables are initialized they contain “garbage” values. You may begin single-stepping through each line of your code by typing “`s`” (for “step”) in the debugging window. When your program ends, you may quit the debugger by typing:

**`quit`**

The debugger includes several commands that are useful when debugging code. For more information on using the debugger, type **`help`** at the debug prompt.

**Questions:** Using the debugger, record the values of the `x`, `y` and `z` variables when program execution begins. Why is it always a good idea to always initialize your variables?