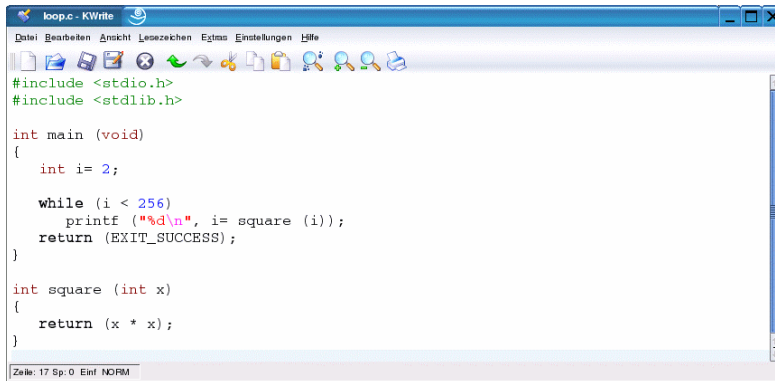


## How to use the GNU Debugger GDB (Basic Usage)

To debug C or C++ programs for the IGW/100 Linux Security Gateway, the GNU tool chain offers a powerful debugger – called GDB (GNU Debugger). This document shows the basic usage.

- **1. Step:** Write a C program with a simple loop and one or more variables. Save the program in a file with the name **loop.c**



```

#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    int i= 2;

    while (i < 256)
        printf ("%d\n", i= square (i));
    return (EXIT_SUCCESS);
}

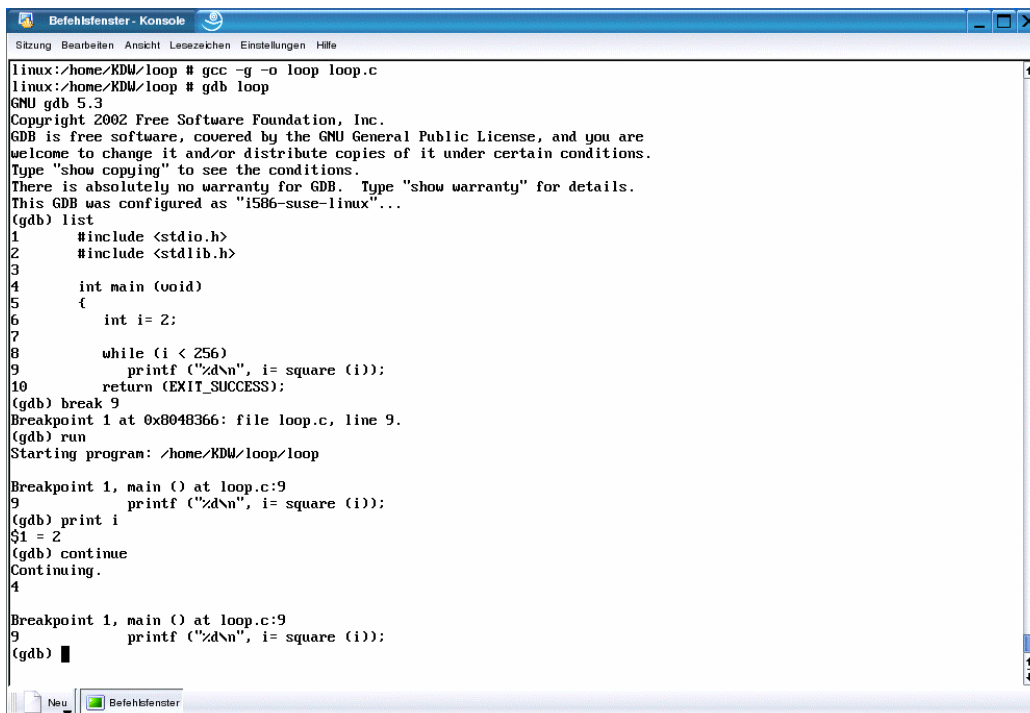
int square (int x)
{
    return (x * x);
}

```

- **2. Step:** Build an executable. Use the GCC with the parameter **-g**. This parameter tells the GCC to place debugging information's within an executable code file.

```
gcc -g -o loop loop.c
```

- **3. Step:** Run GDB. The debugger needs the name of the executable as command line parameter. List the code, set breakpoints and run the program.



```

linux:/home/KDW/loop # gcc -g -o loop loop.c
linux:/home/KDW/loop # gdb loop
GNU gdb 5.3
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i586-suse-linux"...
(gdb) list
1      #include <stdio.h>
2      #include <stdlib.h>
3
4      int main (void)
5      {
6          int i= 2;
7
8          while (i < 256)
9              printf ("%d\n", i= square (i));
10         return (EXIT_SUCCESS);
(gdb) break 9
Breakpoint 1 at 0x0048366: file loop.c, line 9.
(gdb) run
Starting program: /home/KDW/loop/loop

Breakpoint 1, main () at loop.c:9
9      printf ("%d\n", i= square (i));
(gdb) print i
$1 = 2
(gdb) continue
Continuing.
4

Breakpoint 1, main () at loop.c:9
9      printf ("%d\n", i= square (i));
(gdb) █

```

A basic GDB session needs only some simple commands. The first command (**list**) in the following sample lists the C source code within GDB (the debugger knows the line numbers of each C statement).

```
list
break 9
run
:
:
print i
:
:
continue
```

The **break** command sets a breakpoint to the C source code line number 9. **Run** starts the program execution within GDB.

The debugger stops at each breakpoint. With the **print** command it is possible to display the current value of a variable (i.e. **print i** shows the value of the variable “i”). The **continue** statement tells GDB to continue with the program execution.

That’s all.