

Remote-debugging under Linux

The embedded Linux-Starterkits from the DIL/NetPC allow the remote debugging under Linux by using an Ethernet-connection. In the subdirectory \DNPX\TOOLS of the Starterkit CD-ROM there is an approx. 14-kByte large binary file named `gdbserver` stored. This program must be available in the directory `/bin` of the DIL/NetPC. If you use FTP for the data transfer you should know that you can change the execution rights on a later point of time by using the Linux command:

```
chmod +x gdbserver
```

`gdbserver` can also be placed permanently into the FLASH-copy of the root-data system (RIMAGE.GZ). On the development system is the Linux-debugger `gdb` necessary. This will be automatically installed together with the GNU tools. Figure 1 shows the fundamental relations during the remote debugging under Linux.



Figure 1: Overview about the remote debugging

The Figure 2 shows you an example of a Linux debugger session with `gdb` and `gdbserver`. This example was made by the source code from listing 1. It is to be considered, that a C source code has to be translated with the `gcc`-parameter `-g` if the debugger is supposed to use. By that, additional symbol information was transmitted into the file with the executable binary code. Among others this information will be used to display the source code and for setting breakpoints.

If you have never worked with a debugger like `gdb` yet, you should clarify the fundamental working method for yourself before you go on. `gdb` is a so-called source code debugger. This means, that the debugger knows the source code (in this case the C program) line by line. The debugger also knows the names of all variables and functions. Therefore you do not have trouble with storage addresses and so on during the debug procedure.

The fundamental sequence by using a debugger is simple: You set breakpoints into the program and the debugger stops the program execution on these points. At such a breakpoint, the current values will be checked and modified if required. A breakpoint can e.g. be set on a specific source code line because the debugger already knows the source code. The variables are also accessible by name.

A program will be executed by the debugger from breakpoint to breakpoint in real time. The access to the variables is possible on the breakpoints.

The image shows two terminal windows. The left window, titled 'Konsolle', is a telnet client on the development system. It shows a connection to 192.168.0.126, login as 'gast', and the execution of 'ls -al' and 'gdbserver' commands. The right window, titled 'Konsolle <2>', is the embedded system's debugger. It shows the target being set to the remote system, a breakpoint being set at line 9 of main.c, and the program being stepped through. The debugger output shows the current value of 'i' as 2 and the user setting it to 3. The program eventually exits normally.

Figure 2: Debugger session with *gdb* and *gdbserver*

To test this, first of all you have to translate the source code from listing 1 under the name *debugdemo* with *gcc*. This has to be done on the development system. Please make sure that you do not forget the parameter *-g*. The binary file with the executable program will be transmitted to the RAM-disk of the embedded system via FTP and then stored. Now *gdbserver* can be activated.

```

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:
11:     return (EXIT_SUCCESS);
12: }
13:
14: int square (int x)
15: {
16:     return (x * x);
17: }

```

Listing 1: C source code *debugdemo* for the debugger test

To start *gdbserver* on the embedded system, you have to switch into the directory that contains the program *debugdemo* which you would like to test.

Please enter the following command line via Telnet-client.

```
gdbserver 192.168.0.1:2222 ./debugdemo
```

The number 192.168.0.1 is the IP address of the development system and the number 2222 a randomly chosen port number for the TCP/IP-stack. After starting *gdbserver* on the embedded system, the connection to the embedded system can be build up. The following *gdb*-command line is needed:

```
target remote 192.168.0.126:2222
```

In this command line 192.168.0.126 work as IP address of the embedded system and 2222 build the port number. This port number has to be identical to the port number in the line where *gdbserver* were started. Both inputs can be recognized in figure 2. Now you should have activated two windows on the development system: In one window work the Telnet-Client with an active connection to the embedded system. The other window includes the *gdb*-session. The figure 2 shows this status.

Command	Function
<code>gdb debugdemo</code>	start of <i>gdb</i> on the development system
<code>list 1,17</code>	source code output from listing 1 in the debugger
<code>break 9</code>	sets a breakpoint onto the C line number 9
<code>info breakpoints</code>	shows an overview about the breakpoints
<code>delete 1</code>	delete the breakpoint with the number 1
<code>continue</code>	continue program execution
<code>whatis i</code>	determining the type of a variable
<code>print i</code>	output from the content of a variable
<code>set variable i=16</code>	value assignment for a variable
<code>quit</code>	exit <i>gdb</i>

Table 1: Some *gdb*-commands for the remote debugging

Important note for all who have already worked with *gdb*: It is no `run`-command necessary to start the program, which you will test. This program already runs in a waiting state on the embedded system. You can leave this status by use of the `continue`-command.

Figures

Figure 1: Overview about the remote debugging

Figure 2: Debugger session with *gdb* and *gdbserver*

Tables

Table 1: Some *gdb*-commands for the remote debugging

Contact

SSV Embedded Systems
Heisterbergallee 72
D-30453 Hannover
Tel. +49-(0)511-40000-0
Fax. +49-(0)511-40000-40
E-mail: sales@ist1.de
Internet: www.ssv-embedded.de

Document History (Emblinx10e.doc)

Revision	Date		Name
1.00	29.08.2001	First version.	KDW

This document is meant only for the internal application. The contents of this document can change any time without announcement. There is taken over no guarantee for the accuracy of the statements. Copyright © **SSV EMBEDDED SYSTEMS 2001**. All rights reserved.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY KIND. The user assumes the entire risk as to the accuracy and the use of this document.