

【Linux 学习系列十四：使用 gdb 和 gdbserver 构建在线调试环境】

2019-10-04

版 本 历 史

版本	作者	参与者	日期	备注
V1.0	Topsemic		2019/10/04	创建

目录

1.引言	4
2.环境介绍	4
2.1.硬件	4
2.2.软件	4
3. Buildroot 配置	5
4.新建测试程序	7
5.在线调试	8
6.结束语	11

1.引言

单片机一般使用 Jlink 通过 SWD 或者 JTAG 接口直接在 IDE 中在线调试，Linux 应用程序通常是加 printf 输出 log 去调试，这种方式简单，但是有些隐藏的程序 bug 只通过加打印信息不那么容易定位，这时可以通过类似单片机调试的 gdb 调试来实现，本篇为大家介绍 linux 环境下在线调试环境的搭建，希望对大家有所帮助。

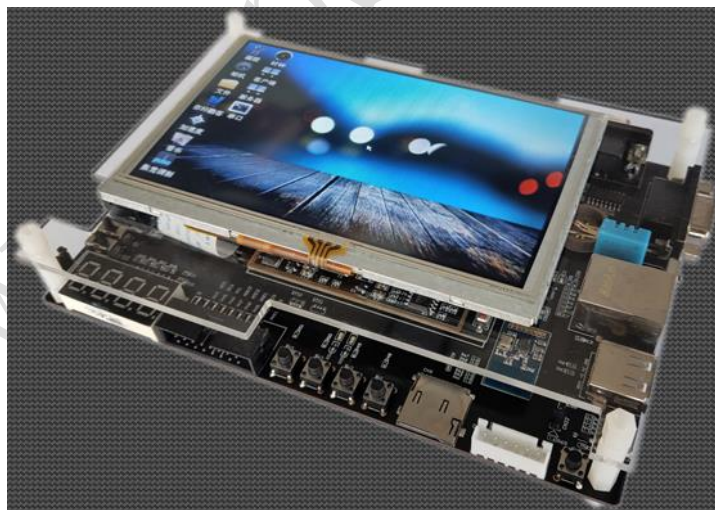
GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed.

它的工作原理是：在主机 Ubuntu 下运行 gdb，在嵌入式板子上运行 gdbserver，这样就可以在线调试了。

2.环境介绍

2.1.硬件

- 1) 网上的一个第三方做的 NUC972 开发板：



有兴趣购买的朋友，可以去他们的淘宝店购买：

<https://s.click.taobao.com/X8mza8w>

2.2.软件

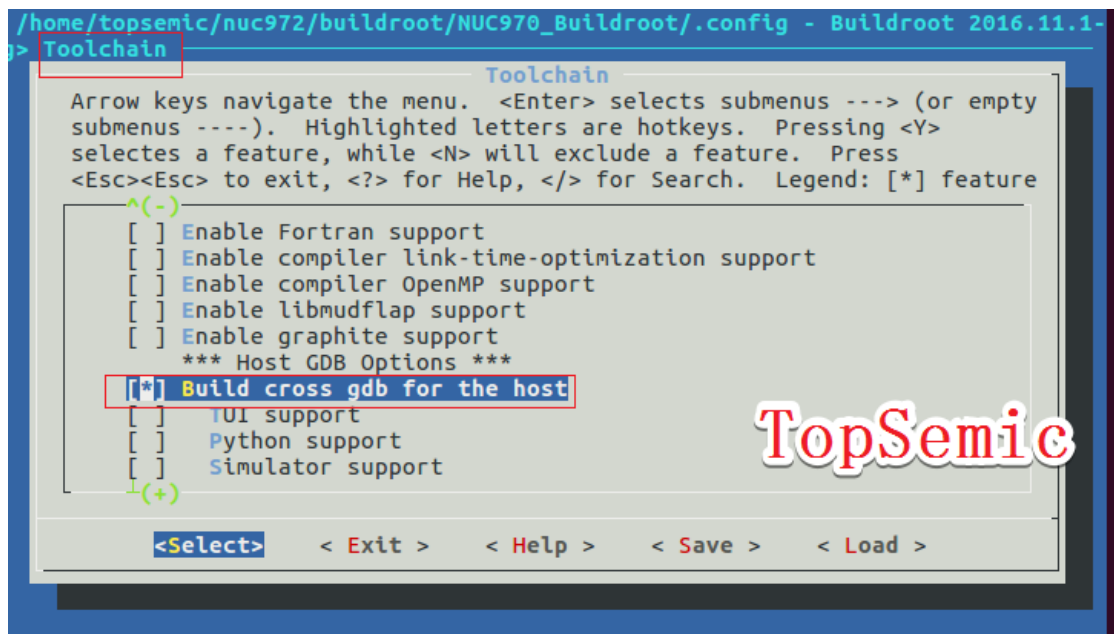
- 1) Uboot 继续使用之前文章用的，无须改动。

- 2) Kernel 在上一篇基础上，无须改动。
- 3) Rootfs 在上一篇用 Buildroot 生成的基础上，需要做一定的改动，用来生成 gdbserver。

3.Buildroot 配置

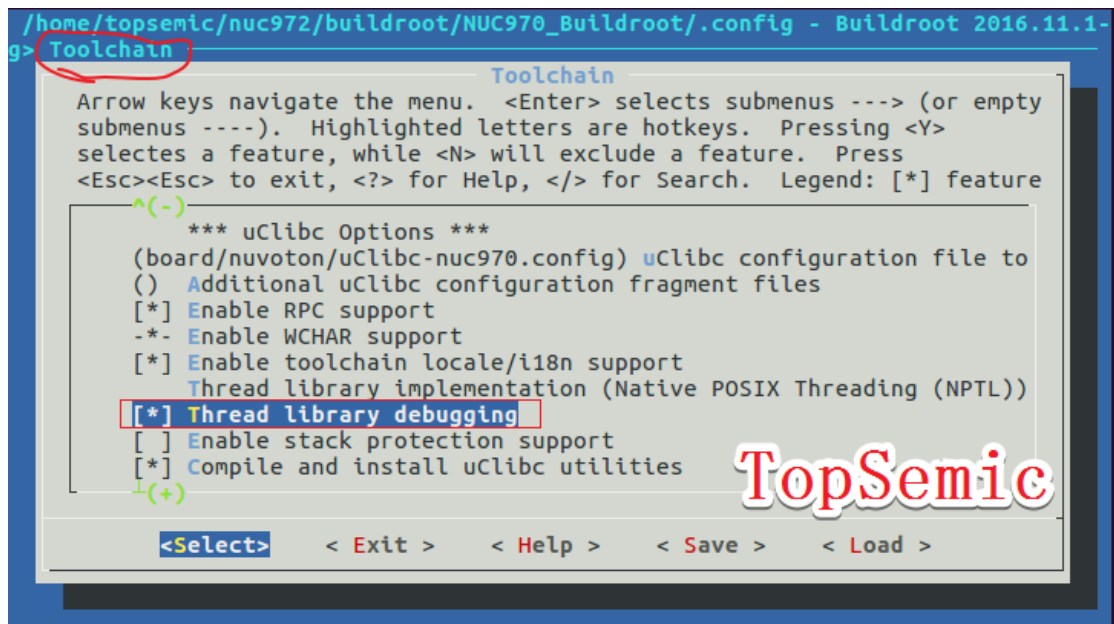
Buildroot 里需要做一定的配置，用来生成 gdb 和 gdbserver，步骤如下：

- 1) 确认 Toolchain | Build cross gdb for the host 是否选中，这个默认是选中的。

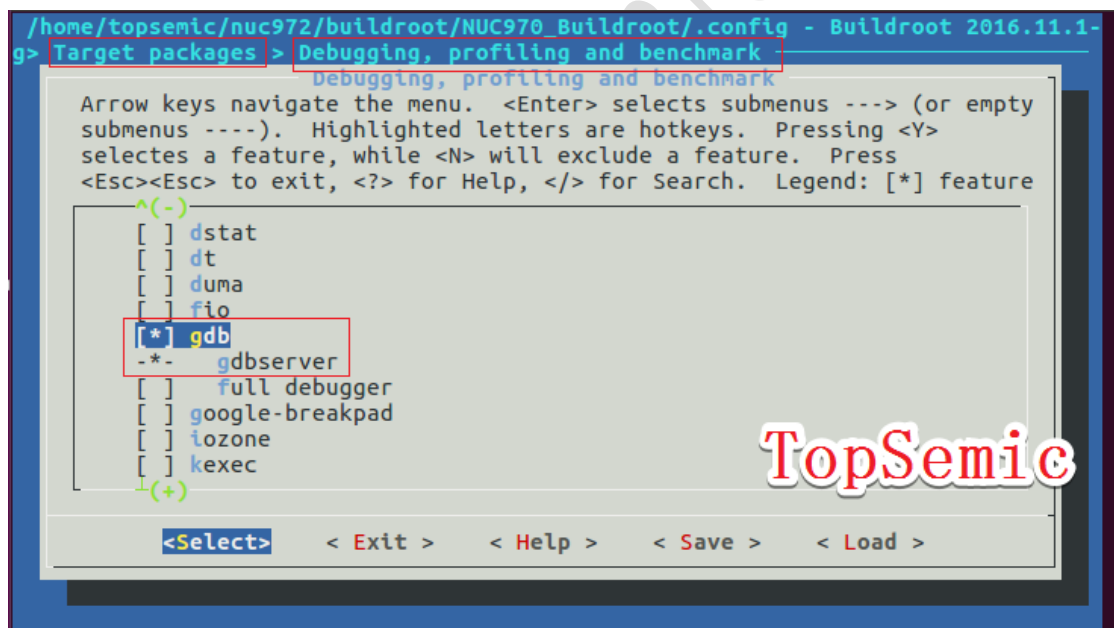


这个的作用是：Build a cross gdb that runs on the host machine and debugs programs running on the target. It requires 'gdbserver' installed on the target。

- 2) 选中 Toolchain 下的 Thread library debugging，注意一定得先选中这个，不然第三步无法执行。



- 3) 选中 Target packages | Debugging, profiling and benchmark->gdb 和
gdbserver



上面的作用是：

This option allows to build gdbserver and/or the gdb debugger for the target.

For embedded development, the most common solution is to build only

'gdbserver' for the target, and use a cross-gdb on the host.

- 4) 保存，编译即可。

生成的 gdb 位于：

`/home/topsemic/nuc972/buildroot/NUC970_Buildroot/output/host/usr/bin`

目录中


生成的 gdbserver 位于:

/home/topsemic/nuc972/buildroot/NUC970_Buildroot/output/target/usr/bin

目录中

- 5) 将上述 gdbserver 直接放到板子的/usr/bin 目录里即可, 然后登录板子输入 gdbserver, 可以看到如下信息, 说明板子的 gdbserver 已经搭建好了。

```
# gdbserver
Usage:  gdbserver [OPTIONS] COMM PROG [ARGS ...]
       gdbserver [OPTIONS] --attach COMM PID
       gdbserver [OPTIONS] --multi COMM
```



COMM may either be a tty device (for serial debugging),
HOST:PORT to listen for a TCP connection, or '-' or 'stdio' to use
stdin/stdout of gdbserver.
PROG is the executable program. ARGS are arguments passed to inferior.
PID is the process ID to attach to, when --attach is specified.

Operating modes:

--attach	Attach to running process PID.
--multi	Start server without a specific program, and only quit when explicitly commanded.
--once	Exit after the first connection has closed.
--help	Print this message and then exit.
--version	Display version information and exit.

Other options:

--wrapper WRAPPER --	Run WRAPPER to start new programs.
--disable-randomization	Run PROG with address space randomization disabled.
--no-disable-randomization	Don't disable address space randomization when starting PROG.

Debug options:

4.新建测试程序

- 1) 新建一个测试程序 gdbtest.c

```
#include <stdio.h>

int main()
{
    char s[64] = "Welcome to www.topsemic.com";

    int a = 1;

    int c = a*2;

    int *ptr = NULL;

    printf("s is :%s\n", s);

    printf("c is : %d\n", c);

    *ptr = 20;
```

```
printf("%d\n",*ptr);  
  
return 0;  
}
```

2) 交叉编译

```
topsemic@topsemic-virtual-machine:~/nuc972/examples/gdbserver$ arm-linux-gcc  
gdbtest.c -o gdbtest -g
```

注: `arm-linux-gcc gdbtest.c -o gdbtest -g` 其中”-g”参数表示进行 GDB 编译。

这个程序放到板子里运行结果如下:

```
# ./gdbtest  
s is :welcome to www.topsemic.com  
c is : 2  
Segmentation fault
```

TopSemic

我们用下面的在线调试方法去看看什么原因导致的 Segmentation fault

5.在线调试

调试前, 将板子和 PC 之间通过网线相连接, 步骤如下:

1) 在开发板可执行程序所在的目录下, 执行如下命令启动 gdbserver:

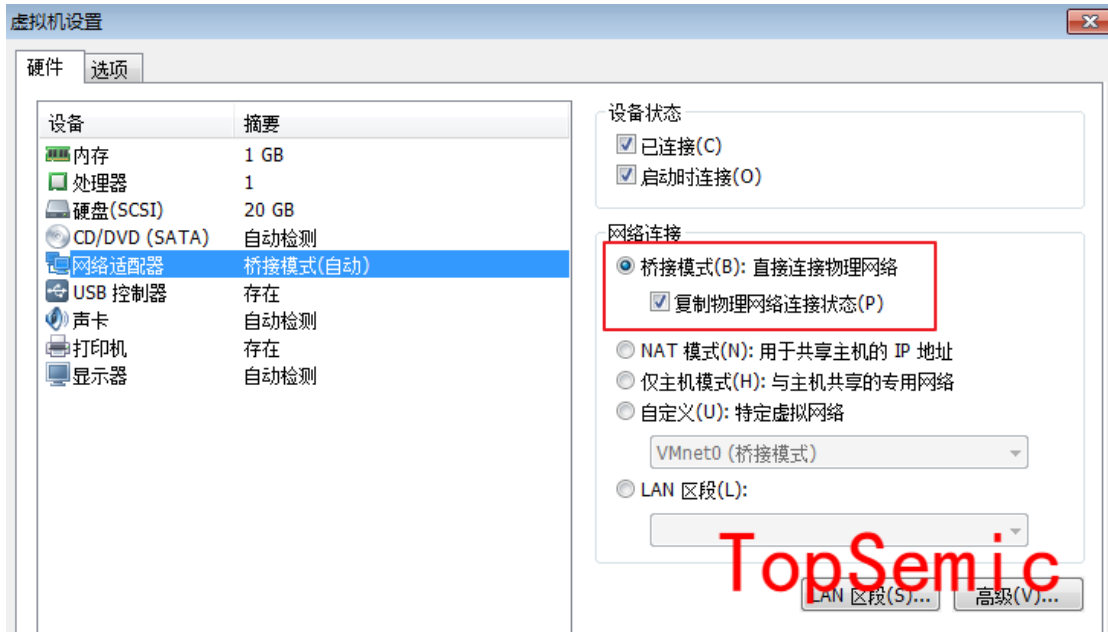
```
# gdbserver 192.168.0.80:1234 ./gdbtest  
Process ./gdbtest created; pid = 557  
Listening on port 1234
```

TopSemic

命令格式: `gdbserver <Host_IP>:<Ports><Program><Arguments...>`

192.168.0.80 为 Ubuntu 的 IP 地址, 1234 为连接的端口号


注: 需要先将虚拟机 Ubuntu 的 IP 配置为固定的 192.168.0.80, 这个设置方法在《Linux 学习系列八: 操作网口》中有介绍



2) 在 Ubuntu 下启动 gdb 调试, 命令格式:

<GDB 可执行程序路径> <应用程序路径>

```
topsemic@topsemic-virtual-machine:~/nuc972/examples/gdbserver$ /home/topsemic/nuc972/buildroot/NUC970_Buildroot/output/host/usr/bin/arm-linux-gdb gdbtest
```

```
topsemic@topsemic-virtual-machine:~/nuc972/examples/gdbserver$ /home/topsemic/nuc972/buildroot/NUC970_Buildroot/output/host/usr/bin/arm-linux-gdb gdbtest
GNU gdb (GDB) 7.10.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=arm-nuvoton-linu
x-uclibcgnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from gdbtest...done.
(gdb) 
```

3) 在弹出的上述对话框(gdb)后输入以下命令，连接开发板

```
(gdb)target remote 192.168.0.100:1234
```

```
(gdb) target remote 192.168.0.100:1234
Remote debugging using 192.168.0.100:1234
Reading /lib/ld-uClibc.so.0 from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to ac
cess files locally instead.
Reading /lib/ld-uClibc.so.0 from remote target...
Reading symbols from target:/lib/ld-uClibc.so.0...(no debugging symbols found)..
.done.
0xb6fe8f80 in _start () from target:/lib/ld-uClibc.so.0
(gdb)
```

其中 192.168.0.100 是开发板的 IP 地址

4) 之后就可输入如下 GDB 调试命令，其他调试命令的详细用法请输入“help 命令名称”查阅。

命令： l，参看代码。

命令： b main，在 main 处设置断点。

命令： b 9，在第六行设置断点。

命令： c，继续执行。

命令： n，单步执行。

命令： q，退出 gdb。

一直输入 c，直到程序结束。

```
(gdb) l
1      #include <stdio.h>
2      int main()
3      {
4          char s[64] = "Welcome to www.topsemic.com";
5          int a = 1;
6          int c = a*2;
7          int *ptr = NULL;
8          printf("s is :%s\n", s);
9          printf("c is : %d\n", c);
10         *ptr = 20;
(gdb)
11         printf("%d\n", *ptr);
12         return 0;
13     }(gdb) b main
Breakpoint 1 at 0x84ac: file gdbtest.c, line 4.
(gdb) b 9
Breakpoint 2 at 0x8564: file gdbtest.c, line 9.
(gdb) c

Continuing.
Reading /lib/libc.so.0 from remote target...

Breakpoint 1, main () at gdbtest.c:4
4          char s[64] = "Welcome to www.topsemic.com";
(gdb) c
Continuing.

Breakpoint 2, main () at gdbtest.c:9
9          printf("c is : %d\n", c);
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x00008578 in main () at gdbtest.c:10
10         *ptr = 20;
(gdb) c
Continuing.

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
```

单步调试，同时查看板子上打印的信息

```
# gdbserver 192.168.0.80:1234 ./gdbtest
Process ./gdbtest created; pid = 569
Listening on port 1234
Remote debugging from host 192.168.0.80
s is :welcome to www.topsemic.com
c is : 2

Child terminated with signal = 0xb (SIGSEGV)
GDBserver exiting
```

可以看到板子程序执行的过程和 Ubuntu 上加的断点运行的进度一致，另外可以发现是因为 line 10 导致的 Segmentation fault，这样就定位到了出问题的地方。

注：<https://man.linuxde.net/gdb> 可以看到详细的 gdb 命令用法。

6.结束语

本期相关的资料在 https://github.com/TopSemic/NUC972_Linux Lesson14 中
本篇为大家介绍了 Linux 下使用 gdb 和 gdbserver 构建在线调试环境，欢迎

大家多交流，可以在网页下方留言讨论，或者发邮件：Topsemic@sina.com，微信公众号如下，欢迎关注：



也可以加入微信群，与我们一起交流

