Remote Debuging in C

Overview

We will be using gdbserver to remote debug our applications.

Adding gdbserver to the image

Installing using apt-get

> sudo apt-get install gdbserver

Copying to the image

Mount image

From build directory

> sudo ./mount-image run run/rpi-basic-image-raspberrypi.rpi-sdimg.20170719

> cp <path>/gdbserver run/partition_2/usr/bin

> chmod +x run/partition_2/usr/bin/gdbserver

Now **sign** and **burn** the image to an SD Card and start up your pi.

Compiling the Code

Assuming the binary is built using cMake withe debug enabled, it will contain symbol information.

To compile the code by hand you will need to specify the **-ggdb** flag to enable debugging.

> gcc rdTest.c -o rdTest -ggdb

Running the gdbserver

On target machine

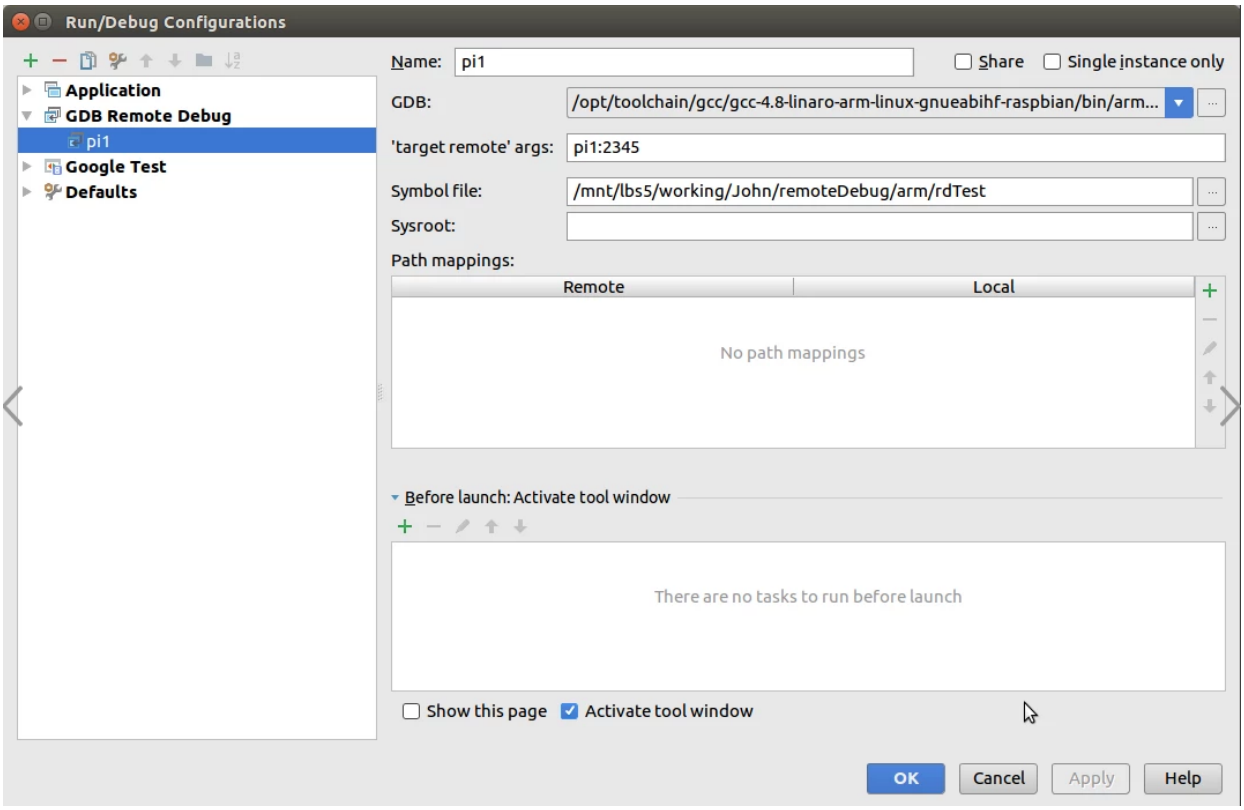
> sudo gdbserver localhost:2345 --attach pid

or

> sudo gdbserver localhost:2345 <executable>

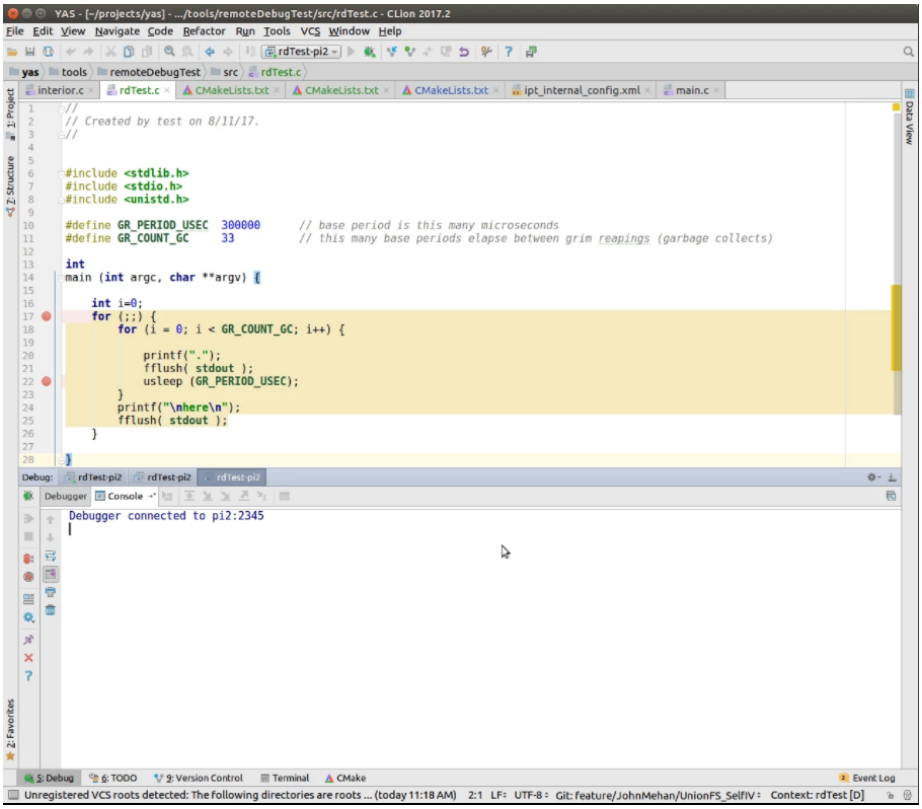
Connecting to the gdbserver from CLion

For CLion, add a GDB Remote Debug Configuration:



Parameter	Description	Example
Name	A name	rdsTest
GDB	Gnu Debugger	/opt/toolchain/gcc/gcc-4.8-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gdb
Target	Specify the remote address	10.10.10.5:2345
Symbol File	Specify the local compiled binary	/home/test/projects/yas/tools/remoteDebugTest/src/rdTest

Hopefully by this point you can connect to the remote gdbserver.



If you get the following error "error while loading shared libraries: libncurses.so.5: cannot open shared object file: No such file or directory"

> sudo apt-get install lib32ncurses5

Troubleshooting/Issues

Connects but never hits a breakpoint

- This can happen if you don't specify the symbol file in Clion

- This can happen if the binary does not contain symbol information.

Verify that the binary has symbol information by issuign the following command:

> nm <binary>

0000000000601058 B __bss_start
0000000000601068 b completed.6973
0000000000601048 D __data_start
0000000000601048 W data_start
0000000000400590 t deregister_tm_clones
...
0000000000601058 D __TMC_END__
U usleep@@GLIBC_2.2.5

- This can happen if anti-debug is enabled.

Testing GDB on Target

Using the following source (lbs5/working/John/remoteDebug/arm/remoteDebugTest/src/rdTest.c)
rdTest.c

```
//
// Created by test on 8/11/17.
//


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

#define GR_PERIOD_USEC  300000          // base period is this many microseconds
#define GR_COUNT_GC             33                      // this many base periods elapse between grim reapings (garbage collects)

int
main (int argc, char **argv) {

    int i=0;
    for (;;) {
        for (i = 0; i < GR_COUNT_GC; i++) {

            printf(".");
            fflush( stdout );
            usleep (GR_PERIOD_USEC);
        }
        printf("\nhere\n");
        fflush( stdout );
    }

}
```

To check if your gdb is working locally you can issue some commands on the target

> gdb rdTest

(gdb) break 24

(gdb) run

Starting program: /usr/bin/rdTest
.....
Breakpoint 1, main (argc=1, argv=0xbefffdb4)
at /home/test/projects/yas/tools/remoteDebugTest/src/rdTest.c:24
24 /home/test/projects/yas/tools/remoteDebugTest/src/rdTest.c: No such file or directory.

```
(gdb) print  i
$1 = 33
```