

GDB介绍

■ GDB 是由 GNU 软件系统社区提供的调试工具，同 GCC 配套组成了一套完整的开发环境，GDB 是 Linux 和许多类 Unix 系统中的标准开发环境。

■ 一般来说，GDB 主要帮助你完成下面四个方面的功能：

1. 启动程序，可以按照自定义的要求随心所欲的运行程序
2. 可让被调试的程序在所指定的调置的断点处停住（断点可以是条件表达式）
3. 当程序被停住时，可以检查此时程序中所发生的事
4. 可以改变程序，将一个 BUG 产生的影响修正从而测试其他 BUG(动态修改程序)

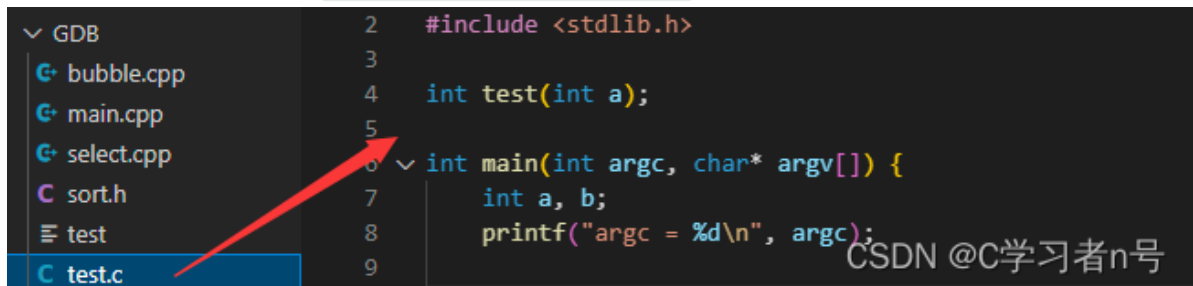
生成调试信息

■ 通常，在为调试而编译时，我们会关掉编译器的优化选项（`-O`），并打开调试选项（`-g`）。另外，`-Wall` 在尽量不影响程序行为的情况下选项打开所有 warning，也可以发现许多问题，避免一些不必要的 BUG。

■ `gcc -g -Wall program.c -o program`或者`gcc program.c -o program -g -Wall`

■ `-g` 选项的作用是在可执行文件中加入源代码的信息，比如可执行文件中第几条机器指令对应源代码的第几行，但并不是把整个源文件嵌入到可执行文件中，所以在调试时必须保证 gdb 能找到源文件。

以下是测试环境，并且通过 `gcc -o test test.c -g -Wall` 生成了可执行文件test。



The screenshot shows a code editor with a file explorer on the left and a C program on the right. The file explorer lists several files: bubble.cpp, main.cpp, select.cpp, sort.h, test, and test.c. The file test.c is selected and highlighted in blue. A red arrow points from the test.c file in the explorer to the main function in the code editor. The code editor shows the following C program:

```
1 #include <stdlib.h>
2
3
4 int test(int a);
5
6 int main(int argc, char* argv[]) {
7     int a, b;
8     printf("argc = %d\n", argc);
9 }
```

The text "CSDN @C学习者n号" is visible in the bottom right corner of the code editor.

GDB启动退出设置参数和帮助

■ 启动和退出

gdb (可执行程序)

输入quit或者q即可退出

■ 给程序设置参数/获取设置参数

set args 10 20

show args

■ GDB 使用帮助

help

- 输入gdb test进入gdb测试程序

```
kiko@Hkiko:~/lesson/GDB$ gdb test
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 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 "x86_64-linux-gnu".
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 test...
(gdb) 
```

CSDN @C学习者n号

- 输入参数，这个参数是给main函数接收的参数。show args用于显示所有的参数

```
(gdb) set args 10 20 30
(gdb) show args
Argument list to give program being debugged when it is started is "10 20 30"
(gdb) 
```

CSDN @C学习者n号

- help可以获取帮助信息，可以通过help + gdb中的一些命令，得到详细信息，比如set。

```
(gdb) help set
Evaluate expression EXP and assign result to variable VAR.
Usage: set VAR = EXP
This uses assignment syntax appropriate for the current language
(VAR = EXP or VAR := EXP for example).
VAR may be a debugger "convenience" variable (names starting
with $), a register (a few standard names starting with $), or an actual
variable in the program being debugged. EXP is any valid expression.
Use "set variable" for variables with names identical to set subcommands.

With a subcommand, this command modifies parts of the gdb environment.
You can see these environment settings with the "show" command.

List of set subcommands:

set ada -- Prefix command for changing Ada-specific settings.
set agent -- Set debugger's willingness to use agent as a helper.
set annotate -- Set annotation_level.
set architecture -- Set architecture of target.
set args -- Set argument list to give program being debugged when it is started.
set auto-connect-native-target -- Set whether GDB may automatically connect to the native target.
--Type <RET> for more, q to quit, c to continue without paging--
```

CSDN @C学习者n号

查看源代码

- 查看当前文件代码
 - list /l (从默认位置显示，如果有多个文件的程序，那么就是main函数所在的文件)
 - list /l 行号 (从指定的行显示)
 - list /l 函数名 (从指定的函数显示)
- 查看非当前文件代码
 - list /l 文件名:行号 --> list select.cpp:2
 - list /l 文件名:函数名 --> list bubble.cpp:bubbleSort
- 设置显示的行数
 - show list / listsize
 - set list / listsize 行数

默认情况下，list直接回车执行，就从程序的启动文件的起始位置开始展示，仅展示10行，此时回车表示继续执行上一次的list，也就是继续往下展示其他的行。

list 行号，这种模式下也是展示10行，行号位置表示展示出来的代码的中间行，而不是起始行。

list 函数名，这种模式下，函数名所在的位置作为展示出来的中间行，和list 行号类似。

断点

- 设置断点
 - b / break 行号
 - b / break 函数名
 - b / break 文件名:行号
 - b / break 文件名:函数
- 查看所有断点信息
 - i / info b/break
- 删除断点
 - d / del / delete 断点编号(可以多个断点编号) --> del 3 4 5
- 设置断点无效
 - dis / disable 断点编号
- 设置断点生效
 - ena / enable 断点编号
- 设置条件断点（一般用在循环的位置）
 - b / break 10 if i==5
- 测试设置断点和查看断点信息

```
(gdb) break 10
Breakpoint 3 at 0x1382: file main.cpp, line 11.
(gdb) break bubble.cpp:12
Breakpoint 4 at 0x1274: file bubble.cpp, line 12.
(gdb) break select.cpp:selectSort
Breakpoint 5 at 0x153f: file select.cpp, line 6.
(gdb) info b
Num      Type           Disp Enb Address            What
3        breakpoint      keep y  0x0000000000001382 in main() at main.cpp:11
4        breakpoint      keep y  0x0000000000001274 in bubbleSort(int*, int) at bubble.cpp:12
5        breakpoint      keep y  0x000000000000153f in selectSort(int*, int) at select.cpp:6
```

CSDN @C学习者n号

- 测试设置断点无效和生效

```
(gdb) info b
Num      Type           Disp Enb Address            What
3        breakpoint      keep y  0x0000000000001382 in main() at main.cpp:11
4        breakpoint      keep y  0x0000000000001274 in bubbleSort(int*, int) at bubble.cpp:12
5        breakpoint      keep y  0x000000000000153f in selectSort(int*, int) at select.cpp:6
(gdb) disable 5
(gdb) info b
Num      Type           Disp Enb Address            What
3        breakpoint      keep y  0x0000000000001382 in main() at main.cpp:11
4        breakpoint      keep y  0x0000000000001274 in bubbleSort(int*, int) at bubble.cpp:12
5        breakpoint      keep n  0x000000000000153f in selectSort(int*, int) at select.cpp:6
```

CSDN @C学习者n号

- 测试设置条件断点

```
(gdb) break 15 if i == 3
Breakpoint 6 at 0x13a6: file main.cpp, line 15.
(gdb) break select.cpp:9 if j == 7
Breakpoint 7 at 0x1564: file select.cpp, line 9.
(gdb) break bubble.cpp:8 if i == 9
Breakpoint 8 at 0x11f8: file bubble.cpp, line 8.
(gdb) i b
Num      Type           Disp Enb Address            What
6        breakpoint      keep y  0x00000000000013a6 in main() at main.cpp:15
        stop only if i == 3
7        breakpoint      keep y  0x0000000000001564 in selectSort(int*, int) at select.cpp:9
        stop only if j == 7
8        breakpoint      keep y  0x00000000000011f8 in bubbleSort(int*, int) at bubble.cpp:8
        stop only if i == 9
```

CSDN @C学习者n号

运行中的调试

- 运行GDB程序
 - start（程序停在第一行）
 - run（遇到断点才停）

- 继续运行，到下一个断点停
c/continue
- 向下执行一行代码（不会进入函数体）
n/next
- 变量操作
p/print 变量名（打印变量值）
ptype 变量名（打印变量类型）
- 向下单步调试（遇到函数进入函数体）
s/step
finish（跳出函数体）
- 自动变量操作
display 变量名（自动打印指定变量的值）
i/info display
undisplay 编号
- 其它操作
set var 变量名=变量值（循环中用的较多，比如直接设置循环变量的值，进行调试）
until（跳出循环）