# gdb常用命令

调用 gdb 编译需要在 cc 后面加 -g 参数再加-o；

[root@redhat home]#gdb 调试文件：启动 gdb

(gdb) l ：（字母 l）从第一行开始列出源码

(gdb) break n :在第 n 行处设置断点

(gdb) break func：在函数 func()的入口处设置断点

(gdb) info break： 查看断点信息

(gdb) r：运行程序

(gdb) n：单步执行

(gdb) c：继续运行

(gdb) p 变量 ：打印变量的值

(gdb) bt：查看函数堆栈

(gdb) finish：退出函数

(gdb) shell 命令行：执行 shell 命令行

(gdb) set args 参数:指定运行时的参数

(gdb) show args：查看设置好的参数

(gdb) show paths:查看程序运行路径；

set environment varname [=value] 设置环境变量。如：set env USER=hchen；

show environment [varname] 查看环境变量；

(gdb) cd 相当于 shell 的 cd;

(gdb)pwd ：显示当前所在目录

(gdb)info program： 来查看程序的是否在运行，进程号，被暂停的原因。

(gdb)clear 行号 n：清除第 n 行的断点

(gdb)delete 断点号 n：删除第 n 个断点

(gdb)disable 断点号 n：暂停第 n 个断点

(gdb)enable 断点号 n：开启第 n 个断点

(gdb)step：单步调试如果有函数调用，则进入函数；与命令 n 不同，n 是不进入调用的函数的

- list ：简记为 l ，其作用就是列出程序的源代码，默认每次显示 10 行。
  - list 行号：将显示当前文件以“行号”为中心的前后 10 行代码，如：list 12
  - list 函数名：将显示“函数名”所在函数的源代码，如：list main
  - list ：不带参数，将接着上一次 list 命令的，输出下边的内容。

注意 ：如果运行 list 命令得到类似如下的打印，那是因为在编译程序时没有加入 -g 选项：

```
(gdb) list
1       ../sysdeps/i386/elf/start.S: No such file or directory.
        in ../sysdeps/i386/elf/start.S
```

- run：简记为 r ，其作用是运行程序，当遇到断点后，程序会在断点处停止运行，等待用户输入下一步的命令。
- 回车：重复上一条命令。
- set args：设置运行程序时的命令行参数，如：set args 33 55
- show args：显示命令行参数
- continue：简讯为 c ，其作用是继续运行被断点中断的程序。
- break：为程序设置断点。
  - break 行号：在当前文件的“行号”处设置断点，如：break 33
  - break 函数名：在用户定义的函数“函数名”处设置断点，如：break cb_button
  - info breakpoints：显示当前程序的断点设置情况

- disable breakpoints Num：关闭断点“Num”，使其无效，其中“Num”为 info breakpoints 中显示的对应值
- enable breakpoints Num：打开断点“Num”，使其重新生效

- step：简记为 s ，单步跟踪程序，当遇到函数调用时，则进入此函数体（一般只进入用户自定义函数）。
- next：简记为 n，单步跟踪程序，当遇到函数调用时，也不进入此函数体；此命令同 step 的主要区别是，step 遇到用户自定义的函数，将步进到函数中去运行，而 next 则直接调用函数，不会进入到函数体内。
- until：当你厌倦了在一个循环体内单步跟踪时，这个命令可以运行程序直到退出循环体。
- finish： 运行程序，直到当前函数完成返回，并打印函数返回时的堆栈地址和返回值及参数值等信息。
- stepi 或 nexti：单步跟踪一些机器指令。
- print 表达式：简记为 p ，其中“表达式”可以是任何当前正在被测试程序的有效表达式，比如当前正在调试 C 语言的程序，那么“表达式”可以是任何 C 语言的有效表达式，包括数字，变量甚至是函数调用。

  - print a：将显示整数 a 的值
  - print ++a：将把 a 中的值加 1,并显示出来
  - print name：将显示字符串 name 的值
  - print gdb_test(22)：将以整数 22 作为参数调用 gdb_test() 函数
  - print gdb_test(a)：将以变量 a 作为参数调用 gdb_test() 函数

- bt：显示当前程序的函数调用堆栈。
- display 表达式：在单步运行时将非常有用，使用 display 命令设置一个表达式后，它将在每次单步进行指令后，紧接着输出被设置的表达式及值。如： display a
- watch 表达式：设置一个监视点，一旦被监视的“表达式”的值改变，gdb 将强行终止正在被调试的程序。如： watch a
- kill：将强行终止当前正在调试的程序
- help 命令：help 命令将显示“命令”的常用帮助信息
- call 函数(参数)：调用“函数”，并传递“参数”，如：call gdb_test(55)
- layout：用于分割窗口，可以一边查看代码，一边测试：

  - layout src：显示源代码窗口

    - layout asm：显示反汇编窗口
    - layout regs：显示源代码/反汇编和 CPU 寄存器窗口
    - layout split：显示源代码和反汇编窗口
    - Ctrl + L：刷新窗口

- quit：简记为 q ，退出 gdb

当然，gdb 的功能远不止这些，包括多进程/多线程/信号/远程调试等功能在这里均没有提及，有需要的读者可以参考其它信息。