一、准备

通过gcc的-g选项生成调试信息。

$gcc -Wall -O2 -g 源文件

-g选项生成GDB使用的一些附加调试信息，这些附加信息可以使GDB工作的更好，但是也可能令其他调试器崩溃或者不能读该文件。

二、GDB基本用法

1、启动:

$gdb 可执行文件名

  2、设置断点:

(gdb) break断点

  程序运行后，到达断点就会自动暂停运行,此时就可以查看该时刻的变量值，显示栈针，重新设置断点或者重新运行等。

断点可以通过函数名，文件内的行号来设置，也可以先指定文件名再指定行号，还可以指定与暂停位置的偏移量，或者用地址来设置。

(gdb) break 函数名

    (gdb) break 行号

    (gdb) break 文件名：行号

    (gdb) break 文件名：函数名

    (gdb) break +偏移量

    (gdb) break -偏移量

    (gdb) break \*地址

在设置断点的时候，如果不指定断点位置，就在下一行代码上设置断点。

  3、条件断点:

    (gdb) break 断点 if 条件

仅在特定条件下中断,对于已存在的断点，可使用condition为其添加条件。

  (gdb) break 断点编号 条件

而删除指定编号断点的触发条件同样使用condition。

    (gdb) condition 断点编号

4、查询断点：

(gdb) info break

5、监视点：

要想找到变量在何处被改变，可以使用watch 命令(监视点watchpoint)。

(gdb) watch <表达式>

    <表达式>发生变化时暂停运行，<表达式>的意思是常量或变量等。

    (gdb) awatch <表达式>

      <表达式>被访问、改变时暂停运行。

    (gdb) rwatch <表达式>

      <表达式>被访问时暂停运行。

6、删除断点和监视点：

(gdb)delete <编号>

7、运行：

(gdb) run 参数

用run命令开始运行，执行run就会执行到设置了断点的位置后暂停运行。经常用到的一个操作是在main()上设置断点，然后执行到main()函数暂停。按照上面的内容，操作命令有：

(gdb) break main

(gdb) run

对于执行到main()函数暂停的操作，可以使用 start 命令达到同样的效果。

(gdb) start

8、显示栈帧：

backtrace 命令可以在遇到断点而暂停执行时显示栈帧。此外，backtrace 的别名还有 where 和 info stack。

(gdb) backtrace 显示所有栈帧

    (gdb) backtrace N 只显示开头N个栈帧

    (gdb) backtrace -N 只显示最后 N 个栈帧

    (gdb) backtrace full不仅显示backtrace，还有显示局部变量

    (gdb) backtrace full N

    (gdb) backtrace full -N

显示栈帧之后，就可以看出程序在何处停止(即断点的位置)，以及程序的调用路径。

9、显示变量：

(gdb) print 变量

10、显示寄存器：

(gdb) info registers

11、单步执行：

执行源代码中一行的命令为next ，执行时如果遇到函数调用，可能想执行到函数内部，此时可以使用 step 命令。

next 命令和 step 命令都是执行源代码中的一行。如果要逐条执行汇编指令，可以分别使用 nexti 和 stepi 命令。

12、继续运行：

调试时，可以使用 continue 命令继续运行程序。程序会在遇到断点后再次暂停运行。如果没有遇到断点，就会一直运行到结束。

(gdb) continue 次数

指定次数可以忽略断点。例如，continue 5 则 5 次遇到断点不停止，第6次遇到断点时才暂停执行。