gdb命令调试需要gcc编译时加上-g 的选项：

gcc –g gdb.c –o gdb

gdb:

**watch：**

**list 或 l表示显示要调试的代码**

l 1表示显示前10行， l命令一次只能列10行；

**继续l或空格键**

**b number（使用断点）**

在另一文件中使用断点；

b xxxxx.c:行数

**delete**用法：delete [breakpoints num] [range...]  
delete可删除单个断点，也可删除一个断点的集合，这个集合用连续的断点号来描述。  
例如：  
delete 5  
delete 1-10

**run（重新运行）**

**step（运行控制）**

**clear**用法:clear   
    删除所在行的多有断点。  
    clear location  
clear 删除所选定的环境中所有的断点  
clear location location描述具体的断点。  
例如：  
clear list\_insert         //删除函数的所有断点  
clear list.c:list\_delet   //删除文件：函数的所有断点  
clear 12                  //删除行号的所有断点  
clear list.c:12           //删除文件：行号的所有断点

**step**

进入函数中去

**continue**

命令c可以使程序继续向下运行

**call**

调用函数

**until**

跳出循环

**info**

1. info args  
   打印出当前函数的参数名及其值。  
   2）info locals  
   打印出当前函数中所有局部变量及其值。  
   3）info catch  
     
   打印出当前的函数中的异常处理信息。  
     
   4）源代码的内存  
   你可以使用info line命令来查看源代码在内存中的地址。info line后面可以跟“行号”，“函数名”，“文件名:行号”，“文件名:函数名”，这个命令会打印出所指定的源码在运行时的内存地址，如：  
   (gdb) info line tst.c:func  
   Line 5 of "tst.c" starts at address 0x8048456 <func+6> and ends at 0x804845d <func+13>.  
     
   5）info break   
   查看断点信息。  
     
   6）info threads  
   看正在运行程序中的线程信息  
     
   7）info registers  
   查看寄存器的情况。（除了浮点寄存器）  
     
   8）info all-registers  
   查看所有寄存器的情况。（包括浮点寄存器）  
     
   9）info registers <regname >  
   查看所指定的寄存器的情况。

**print**

x 按十六进制格式显示变量。  
d 按十进制格式显示变量。  
u 按十六进制格式显示无符号整型。  
o 按八进制格式显示变量。  
t 按二进制格式显示变量。  
a 按十六进制格式显示变量。  
c 按字符格式显示变量。  
f 按浮点数格式显示变量。

(gdb) p i  
$21 = 101

(gdb) p/a i  
$22 = 0x65

(gdb) p/c i  
$23 = 101 'e'

(gdb) p/f i  
$24 = 1.41531145e-43

(gdb) p/x i  
$25 = 0x65

如果要查看地址的话，直接&，就行了

(gdb) p/t i  
$26 = 1100101