GDB

> - **GDB**(GNU Debugger)是一个用来调试C/C++程序的功能强大的调试器，是Linux系统开发C/C++最常用的调试器。
> - **GDB**可以调试C、C++、Go、java、 objective-c、PHP等语言。
> - 虽然它是命令行模式的调试工具，但是它的功能强大到你无法想象，能够让用户**在程序运行时观察程序的内部结构和内存的使用情况。**

GDB主要功能

- 设置**断点(断点可以是条件表达式)**
- 使程序在指定的代码行上暂停执行，便于观察
- **单步执行程序，便于调试**
- 查看程序中**变量值的变化**
- **动态改变程序的执行环境**
- **分析崩溃程**序产生的core文件

目录

一、调试命令参数

> **GDB** 调试的主要参数

1、主要参数

命令	功能
help(h)	查看命令帮助(**help + 命令**)
run(r)	重新开始运行文件
start	单步执行，执行程序，停在第一行执行语句
list(l)	查看源代码(**list-n**:从第n行开始查看，**list+ 函数名**：查看函数)
set	设置变量值
next(n)	单步调试 (逐过程，函数直接执行)
step(s)	单步调试（逐语句，进入函数内执行）
backtrace(bt)	查看函数的调用栈帧和层级的关系
frame(f)	切换函数的栈帧
info(i)	查看函数内部局部变量的数值
finish	结束当前函数，返回函数的调用点
continue(c)	继续执行
print(p)	打印值及地址
quit(q)	退出gdb

2、断点参数

命令	功能
break(b)	在第num行设置断点 (**b+num**)
info breakpoints [n]	查看当前设置的所有断点
delete [breakpoints] [num] (d)	删除第几个断点 （**d+num**）
display	追踪查看具体的变量值
undisplay	取消追踪查看具体的变量值
watch	设置观察点变量发生修改时，打印显示
i watch	显示观察点
enable [breakpoints]	启用断点
disable [breakpoints]	禁用断点
run argv[1] argv[2]	调试时命令行传参

3、提示

> **要调试程序，必须在编译时加上 -g 的参数**
>
> ```
> g++ -g main.cpp -o main
> ```

判断文件是否具有可调试信息

> **gdb + 可执行文件名**
>
> ```
> gdb main
> gdb main -q #
> ```
>
> 1. 可调式
>
>  ```
>  Reading symbols from /home/maxin/Code/cpp/study/main...done.
>  ```
>
> 2. 不可调试
>
>  ```
>  Reading symbols from /home/maxin/Code/cpp/study/main...(no debugging
>  symbols found)...done.
>  ```

二、实战

1、编译并开始调试

```
#include <iostream>
using namespace std;

int main(int args,char **argv)
{
    int N=100;
    int sum=0;
    int i=1;

    while(i<=N)
    {
        sum=sum + i;
        ++i;
    }

    cout<<"sum -->"<<sum<<endl;
    cout<<"over"<<endl;
}
```

编译

> ```
> g++ test.cpp -g -o test_with_g
> ```

开始调试

```
gdb test_with_g
```

简化软件信息

```
gdb test_with_g -q
```

2、调试

> 提示
>
> 直接按下回车执行上一步命令