Linux调试工具GDB

GDB = GNU debuger

功能

- 设置断点
- 更改程序流向
- 监视程序运行时数据（变量、内存等）
- 监视其他信息（寄存器、环境变量等）
- 程序运行时动态修改数据（变量值、信号量）
- 调用回溯
- core dump分析

GDB工作原理

GDB 本身也是一个应用程序，当利用gdb program（program是被调试的目标程序）命令启动gdb程序后。

- GDB本地调试
- GDB远程调试

GDB调试，见代码

gdbserver远程调试

gdb调试器提供了两种不同的远程调试方法

1. stub（插桩）方式
2. gdbserver方式

gdbserver远程调试

用gdb+gdbserver的方式调试嵌入式平台上的Linux应用程序

- 安装arm-linux-gdb
- 安装gdbserver
- 远程调试

库打桩机制

- Linux连接器有强大的打桩机制，允许对共享库代码进行截取、以执行自己的代码或者加入调试信息
- Linux支持在程序的编译阶段、链接阶段或者运行时对库进行打桩
- 编译阶段需要访问源码
- 链接阶段打桩需要访问可重定位文件
- 运行时打桩无以上要求

三种方式

1. 编译时打桩

2. 链接时打桩

3. 运行时打桩

---

gdb常见命令

gcc 编译时应该加上-g参数，并且不要加编译优化

1）进入GDB　　#gdb test

test是要调试的程序，由gcc test.c -g -o test生成。进入后提示符变为(gdb) 。

2）查看源码　　(gdb) l

源码会进行行号提示。

如果需要查看在其他文件中定义的函数，在l后加上函数名即可定位到这个函数的定义及查看附近的其他源码。或者：使用断点或单步运行，到某个函数处使用s进入这个函数。

3）设置断点　　(gdb) b 6

这样会在运行到源码第6行时停止，可以查看变量的值、堆栈情况等；这个行号是gdb的行号。

4）查看断点处情况　　(gdb) info b

可以键入"info b"来查看断点处情况，可以设置多个断点；

5）运行代码　　(gdb) r

6）显示变量值　　(gdb) p n

在程序暂停时，键入"p 变量名"(print)即可；

GDB在显示变量值时都会在对应值之前加上"$N"标记，它是当前变量值的引用标记，以后若想再次引用此变量，就可以直接写作"$N"，而无需写冗长的变量名；

7）观察变量　　(gdb) watch n

在某一循环处，往往希望能够观察一个变量的变化情况，这时就可以键入命令"watch"来观察变量的变化情况，GDB在"n"设置了观察点；

8）单步运行　　(gdb) n

9）程序继续运行　　(gdb) c

使程序继续往下运行，直到再次遇到断点或程序结束；

10）退出GDB　　(gdb) q