快乐的学习 已关注

gdb调试常见命令详细总结（附示例操作）

原创 快乐的学习 已于 2024-01-24 15:33:36 修改 阅读量2.7w 收藏 267 点赞数 28 版权

分类专栏： shell 内核 文章标签： linux

一、简介

通过 gdb调试 我们可以监控程序执行的每一个细节，包括变量的值、函数的调用过程、内存中数据、线程的调度等，从而发现隐藏的错误或者低效的代码，程序的调试过程主要有：单步执行，跳入函数，跳出函数，设置断点，设置观察点，查看变量。
本文将主要介绍 linux 下的gdb调试工具常用的命令和具体的使用实例。

二、 调试过程 介绍

2.1 编译程序加参数时生成调试信息

-g 和 -ggdb 都是令 gcc 生成调试信息，但是它们也是有区别的

选项	解析
g	该选项可以利用操作系统的“原生格式（native format）”生成调试信息。GDB 可以直接利用这个信息，其它调试器也可以使用这个调试信息
ggdb	使 GCC为GDB 生成专用的更为丰富的调试信息，但是，此时就不能用其他的调试器来进行调试了 (如 ddx)

-g也是分级别的

选项	解析
g1	级别1（-g1）不包含局部变量和与行号有关的调试信息，因此只能够用于回溯跟踪和堆栈转储之用。回溯跟踪指的是监视程序在运行过程中的函数调用历史，堆栈转储则是一种以原始的十六进制格式保存程序执行环境的方法，两者都是经常用到的调试手段
g2	这是默认的级别，此时产生的调试信息包括扩展的符号

选项	解析
g3	包含级别2中的所有调试信息，以及源代码中定义的宏

2.2 gdb调试常用命令解析

命令	解析
file [filename]	装入想要调试的可执行文件
kill [filename]	终止正在调试的程序
break [file:]function	在(file文件的)function函数中设置一个断点
clear	删除一个断点，这个命令需要指定代码行或者函数名作为参数
run [arglist]	运行您的程序 (如果指定了arglist,则将arglist作为参数运行程序)
bt Backtrace:	显示程序堆栈信息
print expr	打印表达式的值
continue	继续运行您的程序 (在停止之后，比如在一个断点之后)
list	列出产生执行文件的源代码的一部分
next	单步执行 (在停止之后); 跳过函数调用
nexti	执行下一行的源代码中的一条汇编指令
set	设置变量的值。例如：set nval=54 将把54保存到nval变量中
step	单步执行 (在停止之后); 进入函数调用
stepi	继续执行程序下一行源代码中的汇编指令。如果是函数调用，这个命令将进入函数的内部，单步执行函数中的汇编代码
watch	使你能监视一个变量的值而不管它何时被改变
rwatch	指定一个变量，如果这个变量被读，则暂停程序运行，在调试器中显示信息，并等待下一个调试命令。参考rwatch和watch命令
awatch	指定一个变量，如果这个变量被读

命令	解析
Ctrl-C	在当前位置停止执行正在执行的程序，断点在当前行
disable	禁止断点功能，这个命令需要禁止的断点在断点列表索引值作为参数
display	在断点的停止的地方，显示指定的表达式的值。(显示变量)
undisplay	删除一个display设置的变量显示。这个命令需要将display list中的索引做参数
enable	允许断点功能，这个命令需要允许的断点在断点列表索引值作为参数
finish	继续执行，直到当前函数返回
ignore	忽略某个断点制定的次数。例：ignore 4 23 忽略断点4的23次运行，在第24次的时候中断
info [name]	查看name信息
load	动态载入一个可执行文件到调试器
xbreak	在当前函数的退出的点上设置一个断点
whatis	显示变量的值和类型
ptype	显示变量的类型
return	强制从当前函数返回
txbreak	在当前函数的退出的点上设置一个临时的断点(只可使用一次)
make	使你能不退出 gdb 就可以重新产生可执行文件
shell	使你能不离开 gdb 就执行 UNIX shell 命令
help [name]	显示GDB命令的信息，或者显示如何使用GDB的总体信息
quit	退出gdb

2.3 gdb调试常用参数解析

参数	解析
-e	指定可执行文件名

参数	解析
-c	指定coredump文件
-d	指定目录加入到源文件搜索路径
–cd	指定目录作为路径运行gdb
-s	指定文件读取符号表
-p	指定attach进程

三、具体调试示例讲解

3.1 调试已运行的进程

```
(gdb) gdb    -p        进程名 //对指定进程进行调试
(gdb) gdb    attach    进程名
```

3.2 调试线程

```
(gdb) info    thread    //调试已运行的进程下再列出线程
(gdb) thread    线程号    //切换至线程
```

3.3 查看相关信息

```
(gdb) info    thread      //列出线程
(gdb) info    register    //列出寄存器
(gdb) info    frame       //列出栈帧
(gdb) info    files       //列出当前文件
(gdb) info    share       //列出当前共享库

```

3.4 修改程序执行相关参数

1、程序运行参数

set args 可指定运行时参数。如：

```
(gdb)set args 10 20 30 40 50
(gdb)show args 命令可以查看设置好的运行参数。
```

2、其他参数

参数	描述
path	可设定程序的运行路径
show paths	查看程序的运行路径
set environment varname [=value]	设置环境变量。如：set env USER=hchen
show environment [varname]	查看环境变量
show language	查看当前的语言环境。如果GDB不能识为你所调试的编程语言，那么，C语言被认为是默认的环境。
info frame	查看当前函数的程序语言。
info source	查看当前文件的程序语言。
info breakpoints	显示所有断点
info terminal	显示程序用到的终端的模式

3.5 常用的调试步骤

1、断点的添加

使用break 或者b命令

命令	解析
break function	在进入指定函数时停住
break n	在指定行号停住，如(gdb) break 46
break +offset或break -offset	在当前行号的前面或后面的offset行停住。offiset为自然数。
break filename:linenum	在源文件filename的linenum行处停住。
break filename:function	

命令	解析
break *address	在程序运行的内存地址处停住。
break	break命令没有参数时，表示在下一条指令处停住。

2、断点的删除

命令	解析
删除n号断点	(gdb) delete n
删除所有断点	(gdb) delete
清除行n上面的所有断点	(gdb) clear n

3、程序运行进度调试

（1）连续执行程序，直到遇到断点

```
(gdb)run|r
```

（2）继续执行程序，直到下个断点

```
(gdb) continue|c
```

（3）执行下一行语句

```
(gdb)next|n
```

（4）单步进入

```
(gdb) step|s
```

这样，也会执行一行代码，不过如果遇到函数的话就会进入函数的内部，再一行一行的执行。执行完当前函数返回到调用它的函数

（5）跳出当前函数

```
(gdb) finish
```

这里，运行程序，直到当前函数运行完毕返回再停止。例如进入的单步执行如果已经进入了某函数，可以退出该函数返回到它的调用函数中

（6）跳转指令

```
(gdb) jump location   //指定下一条语句的运行点。可以是文件的行号，可以是file:line格式，可以是+num这种偏移量格式。表式着下一条运行语句位置
```

4、打印程序相关信息

（1） print 命令

输出或者修改指定变量或者表达式的值

```
(gdb) print num
(gdb) p num
(gdb) print file::variable
(gdb) print function::variable
```

其中 file用于指定具体的文件名，funciton 用于指定具体所在函数的函数名，variable表示要查看的目标变量或表达式。
另外，print也可以打印出类或者结构体变量的值。

（2）打印数组

查看一段连续的内存空间的值。比如数组的一段，或是动态分配的数据的大小。你可以使用GDB的“@”操作符，“@”的左边是第一个内存的地址的值，“@”的右边则你你想查看内存的长度。例如，你的程序中有这样的语句：

```
int *array = (int *) malloc (len * sizeof (int));
```

于是，在GDB调试过程中，你可以以如下命令显示出这个动态数组的取值：

```
(gdb) p *array@len
```

@的左边是数组的首地址的值，也就是变量array所指向的内容，右边则是数据的长度，其
保存在变量len中，其输出结果，大约是下面这个样子的：

```
(gdb) p *array@len
$1 = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
```

如果是静态数组的话，可以直接用print数组名，就可以显示数组中所有数据的内容了

（3）源代码显示

命令	解析
list n	显示程序第n行的周围的源程序。
list function	显示函数名为function的函数的源程序。
list +n	显示当前行n后面的源程序。
list -n	显示当前行n前面的源程序。
set listsize	设置一次显示源代码的行数。
show listsize	查看当前listsize的设置。

（4）查看源代码的内存地址

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 and ends at 0x804845d .
```

还有一个命令（disassemble）你可以查看源程序的当前执行时的机器码，这个命令会把目前内存中的指令dump出来。

```
disassemble [Address]
(gdb) disassemble 0x00000000004008fd  //反汇编包含给定地址的整个函数，包括其上方的指令。

disassemble [Start],[End]
(gdb) disassemble 0x000000000040068a,0x00000000004006ac  // 指定要反汇编的起始地址和结束地址

disassemble [Function],+[Length]
(gdb) disassemble main, +10       //指定从给定地址或函数开始反汇编的字节数
```

```
disassemble /m [...]
        指定此选项后，反汇编命令将显示与反汇编指令相对应的源代码行。例如：
(gdb) disassemble /m main
disassemble /r [...]
        当指定此选项时，反汇编命令将显示所有反汇编指令的原始字节值。
(gdb) disassemble /r main
```

（5）查看内存地址保存的值

你可以使用examine命令（简写是x）来查看内存地址中的值。x命令的语法如下所示：

```
(gdb) x/nfu addr
```

n 是一个正整数，表示显示内存的长度，也就是说从当前地址向后显示几个地址的内容。
f 表示显示的格式，参见上面。如果地址所指的是字符串，那么格式可以是s，如果地十是指令地址，那么格式可以是i。
u 表示从当前地址往后请求的字节数，如果不指定的话，GDB默认是4个bytes。u参数可以用下面的字符来代替，b表示单字节，h表示双字节，w表示四字节，g表示八字节。当我们指定了字节长度后，GDB会从指内存定的内存地址开始，读写指定字节，并把其当作一个值取出来。
addr表示一个内存地址。
n/f/u三个参数可以一起使用。例如：

```
(gdb) x/3uh 0x54320 //从内存地址0x54320读取内容，h表示以双字节为一个单位，3表示三个单位，u表示按十六进制显示。
```

（6）查看寄存器

要查看寄存器的值，很简单，可以使用如下命令：

```
(gdb) info registers
```

查看寄存器的情况。（除了浮点寄存器）

```
(gdb) info all-registers
```

查看所有寄存器的情况。（包括浮点寄存器）
寄存器中放置了程序运行时的数据，比如程序当前运行的指令地址（ip），程序的当前堆栈地址（sp）等等。你同样可以使用print命令来访问寄存器的情况，只需要在寄存器名字前加一个$符号就可以了。如：p $eip。

（7）显示当前调用函数堆栈中的函数

```
(gdb) backtrace [-full] [n]
/*命令产生一张列表，包含着从最近的过程开始的所有有效过程和调用这些过程的参数。
n：一个整数值，当为正整数时，表示打印最里层的 n 个栈帧的信息；n为负整数时，那么表示打印最外层n个栈帧的信息；
-full：打印栈帧信息的同时，打印出局部变量的值
注意，当调试多线程程序时，该命令仅用于打印当前线程中所有栈帧的信息。
如果想要打印所有线程的栈帧信息，应执行thread apply all backtrace命令*/
```

（8）frame 命令详解
frame 或 f 会打印出这些信息：栈的层编号，当前的函数名，函数参数值，函数所在文件及行号，函数执行到的语句。
查看当前栈帧中存储的信息

```
(gdb) info frame
Stack level 0, frame at 0x7ffc1da10e80:
 rip = 0x7f800008b4e3 in __epoll_wait_nocancel; saved rip = 0x5560eac8b902
 called by frame at 0x7ffc1da11280
 Arglist at 0x7ffc1da10e70, args:
 Locals at 0x7ffc1da10e70, Previous frame's sp is 0x7ffc1da10e80
 Saved registers:
  rip at 0x7ffc1da10e78
```

该命令会依次打印出当前栈帧的如下信息：
1、当前栈帧的编号，以及栈帧的地址；
2、当前栈帧对应函数的存储地址，以及该函数被调用时的代码存储的地址
3、当前函数的调用者，对应的栈帧的地址；
4、编写此栈帧所用的编程语言；
5、函数参数的存储地址以及值；
6、函数中局部变量的存储地址；
7、栈帧中存储的寄存器变量，例如指令寄存器（64位
堆栈基指针寄存器（64位环境用 rbp表示，32位环境用

4、gdb其他命令用法

（1）搜索源代码

不仅如此，GDB还提供了源代码搜索的命令：

```
(gdb) forward-search     //向前面搜索。
(gdb) reverse-search     //从当前行的开始向后搜索
```

（2）设置观察点(WatchPoint)

观察点一般用来观察某个表达式(变量也是一种表达式)的值是否变化了。如果有变化，马上停住程序。有下面的几种方法来设置观察点：

```
watch    为表达式(变量)expr设置一个观察点。一旦表达式值有变化时，马上停住程序
(gdb) watch i != 10
//这里，i != 10这个表达式一旦变化，则停住。watch <expr> 为表达式（变量）expr设置一个观察点。一量表达式值有变化时，马上停住程序(也是一种断点)。
```

（3）设置捕捉点(CatchPoint)

可设置捕捉点来补捉程序运行时的一些事件。如载入共享库(动态链接库)或是C++的异常。设置捕捉点的格式为：

```
//catch      当event发生时，停住程序
(gdb) info catch
//打印出当前的函数中的异常处理信息。
(gdb) info locals
//打印出当前函数中所有局部变量及其值。
```

（4）强制调用函数

```
(gdb) call <expr>
/*这里,<expr>可以是一个函数，这样就会返回函数的返回值，如果函数的返回类型是void那么就不会打印函数的返回值,但是实践发现，函数运行过程中的打印语句还是没有被打印出来。
表达式中可以一是函数，以此达到强制调用函数的目的。并显示函数的返回值，如果函数返
回值是void，那么就不显示。*/
```

另一个相似的命令也可以完成这一功能——print，print后面可以跟表达式，所以也可以用他来调用函数，print和call的不同是，如果函数返回void，call则不显示，print则显示函数返回值，并把该值存入历史数据中。

（5）终止一个正在调试的程序

```
(gdb) kill
//输入kill就会终止正在调试的程序了。
```

注意：当调试完成后，如果想令当前程序进行执行，消除调试操作对它的影响，需手动将 GDB 调试器与程序分离，分离过程分为 2 步：
1、执行 detach 指令，使GDB调试器和程序分离；
2、执行 quit（或q）指令，退出GDB调试

GDB常用命令集锦 Phoenix_FuliMa 3989

GDB print命令打印字符串全部内容 set print element 0 命令 解释 break NUM 在指定的行上设置断点； 1.break FUNCTION 在某个函数上设置断点。函数重载时，有可能同时在几个重载的函数上设…

GDB调试命令详解_gdb detach 2-7

(gdb)next count //不进入函数内部```- step命令```bash(gdb)step count //进入函数内部``` 1 2 3 4 5 until命令 1、(gdb)until2、(gdb)untillocation 不带参数的 until命令,可以使 GDB调试器快速运行完当前…

gdb常用调试命令_gdb调试命令 2-8

gdb常用调试命令 gdb是GNU项目的一个强大的调试器,可以帮助检查和修改程序的运行状态,发现和修复错误。本文将介绍一些gdb的常用调试命令,以及它们的全称和简写方式。 启动和退出gdb gdb <…

GDB调试 —— 史上最完整 GDB 指令大全 weixin_44798320的博客 5824

目录 1、GDB下载 2、GDB编译 3、GDB下载 4、GDB下载 5、GDB下载 6、GDB下载 1、GDB下载 CentOS $ yum -y install gdb Ubuntu apt-get install gdb 2…

GDB调试器常用命令 LilOne123的博客 6670

Linux GDB调试

GDB常用命令大全_gdb 命令 2-17

GDB常用命令大全 GDB 命令详细解释 一、查看GDB命令帮助 两次按下tab键 然后console 控制台输入 help 二、 GDB是一个强大的命令行调试工具。大家知道命令行的强大就是在于,其可以形成执行…

gdb调试常用指令及案例讲解 明天你好的博客 3842

GDB是一个由GNU开源组织发布的、UNIX/LINUX 操作系统下的、基于命令行的、功能强大的程序调试工具。GDB 支持断点、单步执行、打印变量、观察变量、查看寄存器、查看堆栈等调试手段。…

GDB调试 weixin_44628585的博客 1万+

简单常用gdb命令的学习实操

linux—常用gdb调试命令汇总 weixin_61857742的博客 1万+

如果是函数递归调用，当还没开始递归时，finish会执行完整个函数，自动走完全部递归过程（前提无断点）。因为在linux系统下，默认生成的可执行程序是release版，但是调试需要debug版本。*gd…

GDB常用命令大全 热门推荐 LWJLWJ_的博客 2万+

GDB常用命令大全 GDB 命令详细解释 一、查看GDB命令帮助 两次按下tab键 然后console 控制台输入 help 二、 GDB是一个强大的命令行调试工具。大家知道命令行的强大就是在于，其可以形成执…

嵌入式课件 04-06

ARM9嵌入式系统设计基础教程ppt 第1章 嵌入式系统基础知识 1.1 嵌入式系统的定义和组成 1.1.1 嵌入式系统的定义 1.1.2 嵌入式系统发展趋势 1.1.3 嵌入式系统的组成 1.1.4 实时系统 1.2 嵌入式微处…

arcgis中mdb批量转gdb 08-29

功能：可以在arcgis软件中，利用mdb批量转gdb工具，实现批量的mdb转为gdb

MYSQL中文手册 03-11

言 1. 一般信息 1.1. 关于本手册 1.2. 本手册采用的惯例 1.3. MySQL AB概述 1.4. MySQL数据库管理系统概述 1.4.1. MySQL的历史 1.4.2. MySQL的的主要特性 1.4.3. MySQL稳定性 1.4.4. MySQL表最…

操作系统(内存管理) 09-20

要试着运行这些示例，需要先 复制本代码清单，并将其粘贴到一个名为 malloc.c 的文件中。接下来，我将一次一个部分地对该清单进行解释。 在大部分操作系统中，内存分配由以下两个简单的函数…

MySQL 5.1参考手册中文版 04-28

2.1.1. MySQL支持的操作系统 2.1.2. 选择要安装的MySQL分发版 2.1.3. 怎样获得MySQL 2.1.4. 通过MD5校验和或GnuPG验证软件包的完整性 2.1.5. 安装布局 2.2. 使用二进制分发版的标准MySQL安…

Makefile语法详细总结及示例解析（快速掌握） Luckiers的博客 1万+

makefile可以简单的认为是一个工程文件的编译规则，描述了整个工程的自动编译和链接的规则。 Makefile里主要包含了五个东西：显式规则、隐晦规则、变量定义、文件指示和注释。 （1）显式规…

gdb attach linux 08-27

GDB是一种用来调试程序的工具，可以帮助开发者找到程序中的错误并进行修复。在Linux系统中，可以使用"gdb attach"命令来附加到正在运行的进程上进行调试。 通过"gdb attach"命令，我们可以…

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助

快乐的学习

码龄5年 暂无认证

132	2096	1万+	61万+	
原创	周排名	总排名	访问	等级
2325	1019	618	74	3992
积分	粉丝	获赞	评论	收藏

私信 已关注

搜博主文章

热门文章

SSD硬盘SATA接口和M.2接口区别（详细）总结 58590

gdb调试常见命令详细总结（附示例操作） 27844

芯片行业常用英文术语最详细总结（图文快速掌握） 26781

Makefile语法详细总结及示例解析（快速掌握） 15554

SPEC2006详细参数和测试过程常见问题处理总结（附实例操作） 13616

分类专栏

系统安装 20篇

shell 37篇

git 1篇

笔记 5篇

UEFI 1篇

磁盘 10篇

快乐的学习 已关注

最新评论

Linux下stream内存带宽测试参数和示例…
github_38968457: “STREAM ARRAY_SIZE × 8（双精度） × 3 （三个数组）<= 0 …

Flash闪存储存原理以及NAND flash、N…
highman110: 为什么要这么规定呢?

Flash闪存储存原理以及NAND flash、N…
^_^尽量不坑: 这个芯片就那么做的，可以理解为规定

8b/10b编码方式（详细）总结附实例快速…
苦读: RD+和RD-反了

I2C驱动实例详解
刀不锋马太瘦: 第一张原理图，I2C设备放到总线之上，内核空间以内，毕竟这个是f …

您愿意向朋友推荐“博客详情页”吗?

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

Windows下EDK2快速搭建（详细）过程总结附软件包地址

Linux下基于AHCI controller模块实现SATA Platform驱动附代码详细流程

SATA驱动中FIS命令处理(详细)流程附代码和协议解析

目录

快乐的学习 已关注