第七章 GDB调试器及调试方法

第七章：GDB调试器及调试方法

目标：
本章使学员熟练掌握linux系统下gdb调试器的使用，通过本课的学习，学员应该掌握如下知识：

☑ 了解gdb的作用
☑ 掌握gdb下程序的调试方法

学时：2 学时

教学方法：讲授ppt＋上机操作+实例演示

7.1 gdb介绍

调试器使用：

- 监视程序中变量的值
- 设置断点以使程序在指定的代码行上停止执行
- 单步执行代码

7.2 gdb的使用

- 为使程序能被调试，需要gcc编译时用 -g 选项为程序编译时添加调试信息。

  例如：gcc –g –o helloworld helloworld.c

7.2 gdb的使用

- 在命令行上键入 gdb 并按回车键就可以运行 gdb 了, 如果一切正常的话, gdb 将被启动并且你将在屏幕上看到类似的内容:

```
GNU gdb 6.6-debian
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
(gdb) _
```

7.3 gdb命令

- file　　装入想要调试的可执行文件。
- kill　　终止正在调试的程序。
- list　　列出产生执行文件的源代码的一部分。
- next　　执行一行源代码但不进入函数内部。
- step　　执行一行源代码而且进入函数内部。
- run　　执行当前被调试的程序。
- c　　继续运行程序。
- quit　　终止gdb。
- watch　　使你能监视一个变量的值而不管它何时被改变。
- backtrace　　栈跟踪，查出代码被谁调用。
- print　　查看变量的值。
- 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。

7.3 gdb命令

- whatis　　显示变量或函数类型。
- break　　在代码里设断点，这将使程序执行到这里时被挂起。
- info break　　显示当前断点清单，包括到达断点处的次数等。
- info files　　显示被调试文件的详细信息。
- info func　　显示所有的函数名称。
- info local　　显示当函数中的局部变量信息。
- info prog　　显示被调试程序的执行状态。
- delete [n]　　删除第n个断点。
- disable[n]　　关闭第n个断点。
- enable[n]　　开启第n个断点。
- ptype　　显示结构定义。
- set variable　　设置变量的值。
- call name(args)　　调用并执行名为name，参数为args的函数。
- Finish　　终止当前函数并输出返回值。
- return value　　停止当前函数并返回value给调用者。

7.3 gdb命令

[break命令的使用]

1. 根据行号设置断点:
   (gdb) break linenum
2. 根据函数名设置断点:
   (gdb) break funcname
3. 执行非当前源文件的某行或某函数时停止执行:
   (gdb) break filename:linenum
   (gdb) break filename:funcname
4. 根据条件停止程序执行:
   (gdb) break linenum if expr
   (gdb) break funcname if expr

7.4 基本的gdb调试

```
neusoft@neusoft-desktop:~/mypro/gdbtest$ gcc -g -o main main.c add.c del.c modify.c
neusoft@neusoft-desktop:~/mypro/gdbtest$ gdb main
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
(gdb) list
1       #include "main.h"
2
3       int main()
4       {
5               add();
6               del();
7               modify();
8               return 0;
9       }
(gdb)
```

7.4 基本的gdb调试

```
(gdb) l
1       #include "main.h"
2
3       int main()
4       {
5               int i,j[10];
6
7               for(i=0;i<10;i++)
8                       j[i]=i;
9               add();
10              del();
(gdb) b 8
Breakpoint 1 at 0x804838e: file main.c, line 8.
(gdb) r
Starting program: /home/neusoft/mypro/gdbtest/main

Breakpoint 1, main () at main.c:8
8                       j[i]=i;
(gdb) p i
$1 = 0
(gdb) p j
$2 = {-1076173155, -1208455346, -1207725607, 134518200, -1076178376, 134513312,
  -1207472140, 134518200, -1076178344, 134513705}
(gdb) whatis j
type = int [10]
(gdb) whatis i
type = int
(gdb) whatis main
type = int ()
```

```
(gdb) s
7               for(i=0;i<10;i++)
(gdb) s

Breakpoint 1, main () at main.c:8
8                       j[i]=i;
(gdb) p j
$3 = {0, -1208455346, -1207725607, 134518200, -1076178376, 134513312,
  -1207472140, 134518200, -1076178344, 134513705}
(gdb) p &j[0]
$4 = (int *) 0xbfdad218
(gdb) x/d 0xbfdad218
0xbfdad218:     0
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x0804838e in main at main.c:8
        breakpoint already hit 2 times
(gdb) disable 1
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep n   0x0804838e in main at main.c:8
        breakpoint already hit 2 times
(gdb) delete 1
(gdb) info b
No breakpoints or watchpoints.
(gdb) b 9
Breakpoint 1 at 0x80483a2: file main.c, line 9.
(gdb) r
Starting program: /home/neusoft/mypro/gdbtest/main

Breakpoint 1, main () at main.c:9
9               add();
(gdb) s
add () at add.c:4
4               printf("add\n");
(gdb) bt
#0  add () at add.c:4
#1  0x080483a7 in main () at main.c:9
(gdb)
```

7.5 内存出错的gdb调试

- 常见的段错误

```
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ ./err
Please input a string:qwqwq
段错误
```

- 出错程序代码

```
#include <stdio.h>
#include <string.h>

static char buff[256];
static char* string;

int main()
{
        printf("Please input a string:");
        gets(string);
        printf("\nYou string is:%s\n",string);
}
```

7.5 内存出错的gdb调试

```
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ gdb err
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
(gdb) l
1       #include <stdio.h>
2       #include <string.h>
3
4       static char buff[256];
5       static char* string;
6
7       int main()
8       {
9               printf("Please input a string:");
10              gets(string);
(gdb) r
Starting program: /home/neusoft/mypro/gdbtest/test2/err
Please input a string:123abc

Program received signal SIGSEGV, Segmentation fault.
0xb7f5dc2a in gets () from /lib/tls/i686/cmov/libc.so.6
(gdb) print string
$1 = 0x0
(gdb) b 10
Breakpoint 1 at 0x80483c1: file err.c, line 10.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/neusoft/mypro/gdbtest/test2/err

Breakpoint 1, main () at err.c:10
10              gets(string);
(gdb) set variable string=&buff
(gdb) n
Please input a string:123abc
11              printf("\nYou string is:%s\n",string);
(gdb) n

You string is:123abc
12      }
(gdb)
```

7.5 内存出错的gdb调试

- 利用core文件进行出错文件的调试，使用ulimit –c来设定产生core文件的容量，0为不产生core文件，然后执行通过gdb进行调试。

```
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ ls
err  err.c
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ ./err
Please input a string:12
段错误 (core dumped)
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ ulimit -c 1000
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ ls
core  err  err.c
```

7.5 内存出错的gdb调试

- 利用core文件直接定位出错代码。

```
neusoft@neusoft-desktop:~/mypro/gdbtest/test2$ gdb err core
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /lib/tls/i686/cmov/libc.so.6...done.
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./err'.
Program terminated with signal 11, Segmentation fault.
[New process 20501]
#0  0xb7ed9c2a in gets () from /lib/tls/i686/cmov/libc.so.6
(gdb)
```

7.5 内存出错的gdb调试

- 练习：利用core文件定位以下出错代码。

```
1 #include <stdio.h>
2 int main()
3 {
4     int *p=0;
5     *p=1;
6     return 0;
7 }
```

7.6 其它调试方法

- 利用#ifdef #endif为程序添加调试信息。

```c
#include <stdio.h>

int *p=0;
int q=0;

void aaa()
{
        *p=0;
}

void bbb()
{
        q=1;
}

int main()
{
        bbb();
#ifdef DEBUG
printf("bbb is ok\n");
#endif
        aaa();
#ifdef DEBUG
printf("aaa is ok\n");
#endif
        printf("Hello,World!\n");
        return 0;
}
```

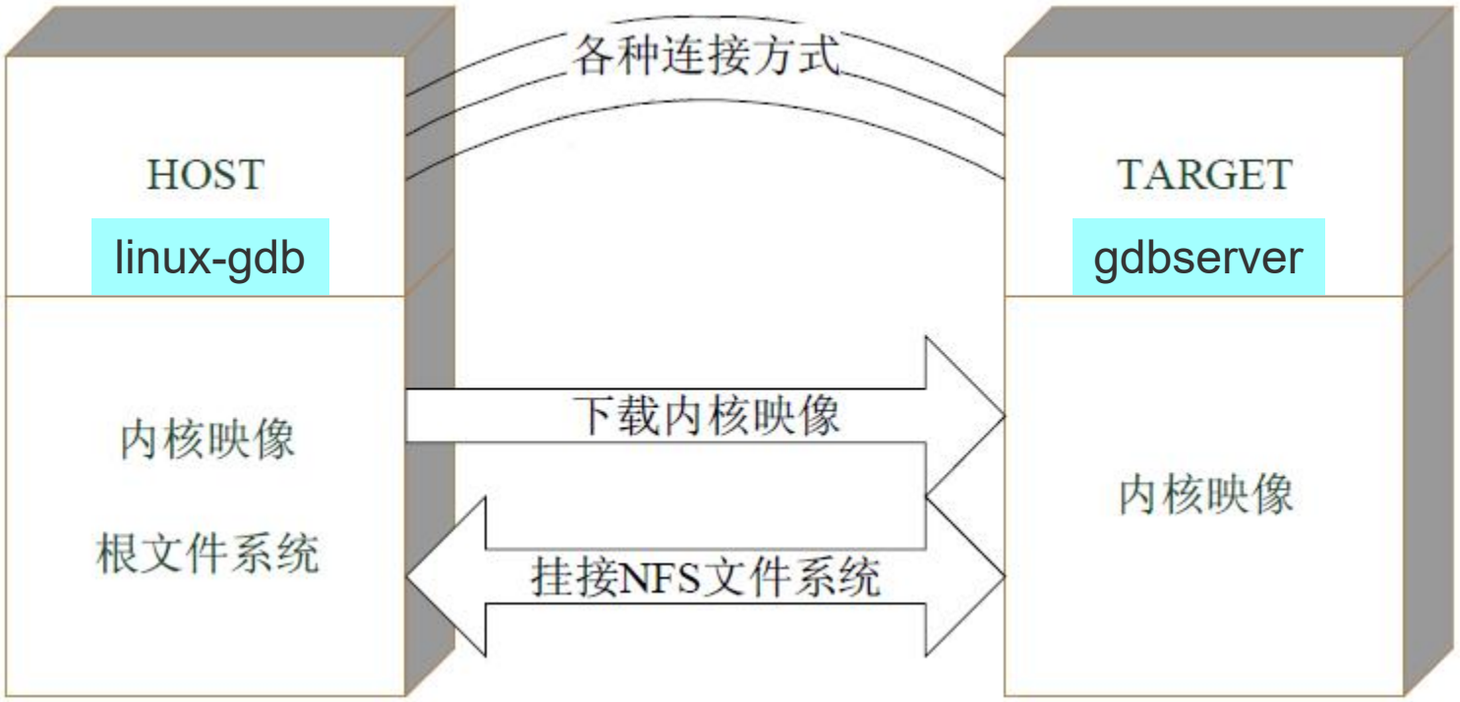
7.6 其它调试方法

- 通过gcc的-D参数进行功能的开启，执行后，通过开启调试信息，并判断出出错函数为aaa。

```
neusoft@neusoft-desktop:~/mypro/gdbtest/test3$ gcc -DDEBUG err.c
neusoft@neusoft-desktop:~/mypro/gdbtest/test3$ ls
a.out  err.c
neusoft@neusoft-desktop:~/mypro/gdbtest/test3$ ./a.out
bbb is ok
段错误 (core dumped)
neusoft@neusoft-desktop:~/mypro/gdbtest/test3$
```

7.7 嵌入式gdb简介

- 在嵌入式交叉开发模型中，gdb+gdbserver是调试目标板的常用方法。
- gdb运行在主机上，而gdbserver运行在目标板上。它们通过网络连接实现远程调试。



7.7 嵌入式gdb简介

- 安装方法
  1.下载gdb源码包，例如gdb-7.4.tar.gz。
  2.解压缩源码包。
  3.编译主机端调试器linux-gdb，并修改环境变量。
  4.编译目标板调试器gdbserver。
  5.下载gdbserver至目标板。

7.7 嵌入式gdb简介

- 调试方法

  主机IP：192.168.0.33，目标板IP：192.168.0.34

  1.用交叉编译器编译程序hello，并拷贝至目标板。

  2.运行目标板上的gdbserver。

  例：./gdbserver 192.168.0.33:6666 hello

  3.运行主机上的linux-gdb，并连接目标板。

  例：linux-gdb hello

  （gdb）target remote 192.168.0.34:6666

  4.调试程序。

  连接后调试方法与本地gdb调试方法相同。