Linux系统及Shell编程

东北林业大学

卢洋

第三章

Linux C 开发工具

1. 编辑器vi/vim的使用

2. 编译器gcc的使用

3. 调试器gdb的使用

4. 工程管理器make的使用

3.3

调试器gdb的使用

断点的设置和管理

debug最为有效的方式

设置断点

1. 以行数设置断点

- 格式: break n
- 功能: 当程序运行到指定行时，会暂停执行，指定行的代码不执行
- 例如:

```
(gdb)break 15

(gdb)run
```

设置断点

2. 以函数设置断点

- 格式: break [函数名]
- 例如:

(gdb)break get_sum

(gdb)run

设置断点

3. 以条件表达式设置断点

- 格式: break [行号或函数名] if [条件]
- 功能: 程序在运行过程中，满足设定条件时，程序在所设置处中断
- 例如:

(gdb)break 7 if i==99

含义: 当程序执行到第7行时，判断条件i==99是否成立，若成立则中断

设置断点

4. 以条件表达式变化设置断点

- 格式: watch [条件表达式]

- 功能: 程序在运行过程中，当满足设定条件时，程序中断

- 注意：watch必须在程序运行的过程中设置观察点，即运行run之后，并且要保证条件表达式中的变量已经使用过。

以条件表达式变化设置断点的实例

- 例1

(gdb)break 13

(gdb)run

(gdb)watch sum==3

- 例2

(gdb)break 5

(gdb)run

(gdb)watch sum==3

查看断点

- 查看当前设置的断点
- 格式: info breakpoints
- 例如:

(gdb)break 7

(gdb)break 15 if res==5050

(gdb)info breakpoints

管理断点

1. 使中断失效或有效

(1) 失效: disable [断点编号]

(2) 有效: enable [断点编号]

2. 删除断点

(1) clear [行号]: 删除此行的断点

(2) delete [断点编号]: 删除指定编号的断点， 若有一次删除多个断点，各断点编号以空格分开。

(3) delete: 删除程序中所有的断点

查看和设置变量的值

- 当程序执行到中断点暂停时，往往需要查看变量或表达式的值，借此了解程序的执行状态，进而发现问题。
- print命令
- 功能: 打印变量或表达式的值，还可以用来对某个变量进行赋值。
- print [变量或表达式]: 打印变量或表达式的值
- print [变量]=[值]: 对变量进行赋值

print命令实例

(gdb)break 7

(gdb)run

(gdb)print i < n

(gdb)print i

(gdb)print sum

(gdb)print i=200

(gdb)continue

查看和设置变量的值

2. whatis

- 功能: 用于显示某个变量或表达式的数据类型
- 格式: whatis [变量或表达式]

whatis命令实例

(gdb)break 7

(gdb)run

(gdb)whatis i

(gdb)whatis sum

(gdb)whatis sum+0.5

查看和设置变量的值

3. set

- 功能: 给变量赋值
- 格式: set variable [变量]=[值]

控制程序的执行

- 当程序执行到指定的中断点时，完成相关的debug操作后，可以让程序继续运行

1. continue

- 程序继续运行，直到下一个断点或运行完毕

2. kill

- 结束当前程序的调试

3. next/step

- 功能: 一次一条执行程序代码
- 区别: next把函数调用当做一条语句来执行；step追踪进入函数，一次一条地执行内部代码。