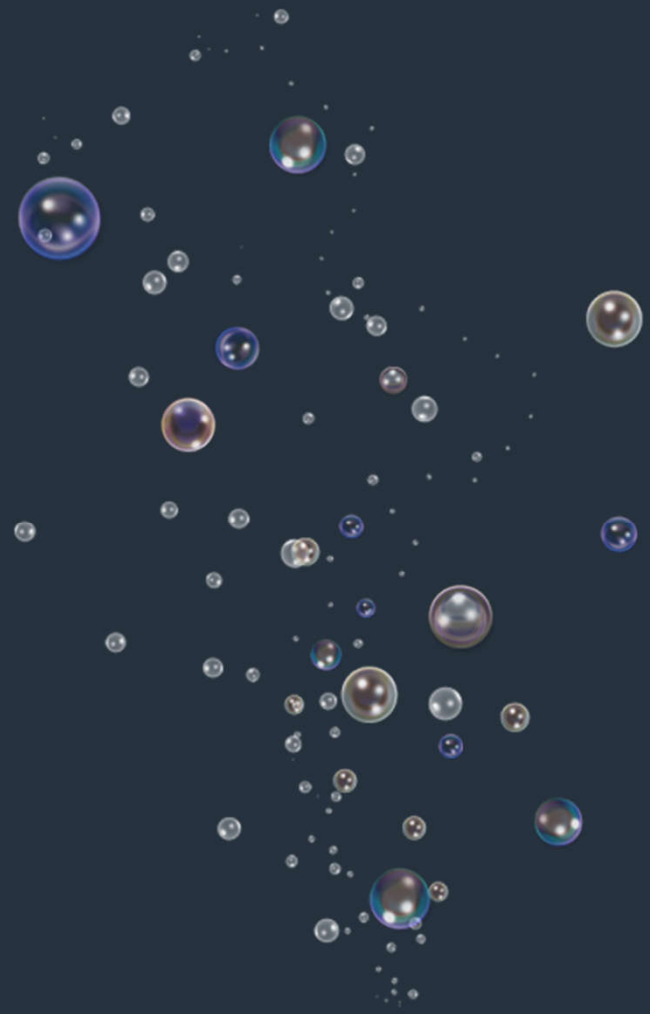


gdb简介

大连理工大学 赖晓晨

gdb简介

- gdb是GNU发布的Linux下的字符界面调试工具，功能强大。
- gdb的主要功能
  - 按照用户的要求启动程序
  - 让被调试的程序在任意断点处停止（断点可以是条件表达式）
  - 程序暂停时可检查运行环境
  - 程序暂停时可动态改变运行环境
- 要使用gdb调试程序，在用gcc编译源文件时要指定-g选项，以使程序中包含必要的信息

gdb使用举例

```
#include <stdio.h>
int func(int n)
{
    int sum=0,i;
    for(i=1;i<=n;i++)
        sum+=i;
    return sum;
}
int main()
{
    int i;
    long result = 0;
    for(i=1;i<=100;i++)
        result+=i;
    printf("result(1~100)= %d\n",result);
    printf("result(1~250)= %d\n",func(250));
    return 0;
}
```

gdbtest.c

gdb的常用命令

- file：装入想调试的可执行文件
- kil：终止正在调试的程序
- list：列出源代码
- next：step over
- step：step into
- finish：step out
- quit：退出gdb

gdb的常用命令（续）

- break：设置断点
- make：不退出gdb就可以重新产生可执行文件
- shell：在gdb中执行shell命令
- print：显示变量值
- display：单步运行时自动显示变量值
- **(gdb) ↙ ：重复执行命令**
- **Tab键：命令补齐**
- **命令缩写：**

在gdb中运行shell程序

- 在gdb中，可以执行linux的shell命令

  格式：shell <command string>

```
例如：(gdb)shell ls
      (gdb)shell pwd
```

调试已经运行的程序

- 在linux下用ps命令查看正在运行的程序的PID，然后执行

```
gdb program PID
```

- 先用gdb program关联上程序，进入gdb后用attach命令挂接进程PID，用detach取消挂接

```
gdb progarm
(gdb)attach PID
```

~/exp/gdb/grun.c

设置断点

- 用break命令设置断点，有以下几种方法：
  - break function　//执行到某函数中止
  - break linenum　//执行到某行中止
  - break +offset　//在当前行后offset行停止
  - break –offset　//在当前行前offset行停止
  - break filename:linenum
  - break filename:function
  - break *address
  - **tbreak　//仅中断一次，中断后断点即自动删除**
  - **break … if condition　//条件断点**

查看断点

- 使用info命令

```
info breakpoint              //查看所有断点
info breakpoint n            //查看第n个断点
缩写:
        i b 2
```

条件断点

- 设置条件断点：

```
break 18 if i==50
```

- 用condition命令可以修改条件：

  condition bnum expression

```
condition 1 i==60
```

- 清除停止条件（变为一般断点）

  condition bnum

```
condition 1
```

条件断点（续）

- ignore命令可以指定程序运行时忽略停止条件几次。

  ignore bnum count

```
ignore 1 20
```

设置观察点

观察点一般用来观察某个表达式是否变化了，如果变化了，则暂停程序。

- watch expr　　//表达式expr变化则停止
- rwatch expr　　//表达式expr被读时停止
- awatch expr　　//表达式expr被读写时停止

设置观察点

- 观察点一般用来观察某个表达式是否变化了，如果变化了，则暂停程序。
  - watch expr　　//表达式expr变化则停止
  - rwatch expr　　//表达式expr被读时停止
  - awatch expr　　//表达式expr被读写时停止
- 查看观察点：
  - info watchpoints
  - info watchpoints n

设置捕捉点

- 可以设置捕捉点用来捕捉程序运行时发生的一些事件，语法为：“catch event”（或“tcatch event”），其中event为：
  - throw：一个C++抛出的异常
  - catch：一个C++捕捉的异常
  - exec：调用系统调用exec时（仅HP-UX）
  - fork：调用系统调用fork时（仅HP-UX）
  - vfork：调用系统调用vfork时（仅HP-UX）
  - load：载入动态链接库时（仅HP-UX）
  - unload：卸载动态链接库时（仅HP-UX）

维护停止点

- 维护停止点的命令
  - delete
  - clear
  - disable
  - enable

为停止点设定运行命令

- 可以利用command命令为停止点设定一系列运行命令。

commands bnum

…command list

end

```
commands 1
shell pwd
shell ls –l
end
```

单步调试

- next：step over
- step：step into
- finish：step out

查看栈信息

- backtrace命令：查看栈信息
- f n：查看第n层栈
- up n：向栈的上方移动n层
- down n：向栈的下方移动n层
- info f：提供更加详细的当前层信息
- info locals：显示当前函数所有局部变量值

查看源程序

- list linenum
- list function
- list
- list -　　//向当前行前显示
- show listsize
- set listsize n

查看源程序（续）

- list first, last
- list , last
- list +　　//向当前行后显示
- x addr　　//查看内存
- info line n　　//显示第n行的内存地址
- disassemble func　　//反汇编函数func

查看运行时数据

- p命令：查看程序中变量、数组的值

```
print /<f> n        //显示变量n的值
print /<f> ::n      //显示全局变量n的值
print /<f> array //显示数组各元素值
print /<f> *array@length
//显示动态分配内存的数组各元素值
```

/<f>表示显示格式

设置自动显示

- 程序停住时，或者单步跟踪时，希望一些变量会自动显示，可使用gdb命令是display。

```
display i                    //显示变量i的值
display $pc                  //显示当前内存地址
```

- 查询自动显示

```
info display
```

删除和禁用自动显示

- 删除自动显示

```
undisplay dnum
delete display dnum
```

- 禁用和启用自动显示

```
disable display dnum
enable display dnum
```

其他可以查看的信息

- 查看gdb的环境变量

```
show convenience
```

- 查看寄存器

```
info register
info all-register    //包括浮点寄存器
```

改变程序执行的流程

- 执行时改变程序变量值

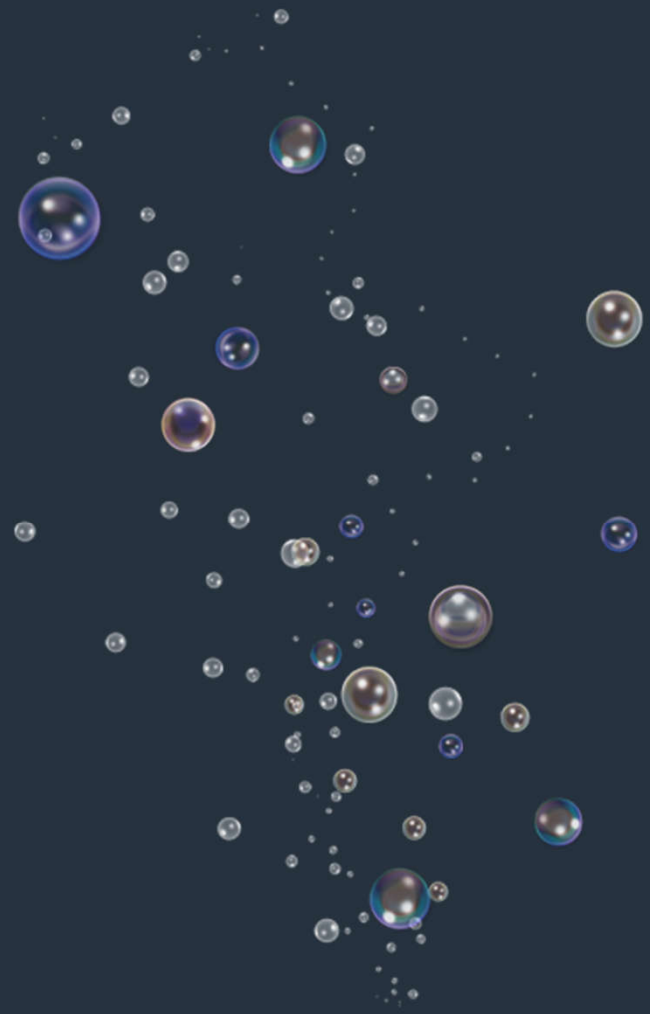
```
print i=10
set var i=15
```

- 跳转执行

```
jump linenum     //跳到某行后执行
jump address     //跳到某内存地址处执行
```

在不同语言中使用gdb

- gdb支持c、c++、fortran、pascal、java、chill、assembly、modula-2。
- show language：查看目前的语言环境
- info frame：查看当前函数的语言
- info source：查看当前文件的语言
- set language：查看gdb支持的语言



嵌入式软件设计

大连理工大学 赖晓晨