

# C++

- [gdb](#) 

# gdb 入门

## 安装GDB

### 安装

```
SH$ gdb program          # program 可执行文件
$ gdb program core        # gdb 可执行文件 core 核心文件
$ gdb program pid         # 通过 pid 调试 D 可执行文件 attach 可执行文件 PATH 可执行文件
```

### 基本命令

```
SH(gdb) r/run             # 运行
(gdb) c/continue          # 继续
(gdb) n/next              # 下一步
(gdb) s/step              # 单步
(gdb) ni/si               # 单步/非侵入式单步
(gdb) fini/finish         # 结束/结束
(gdb) u/util              # 用户交互

(gdb) set args            # 设置 args 10 20 30
(gdb) show args           # 显示 args
(gdb) path <dir>          # 设置路径
(gdb) show paths          # 显示路径
(gdb) set env <name=val> # 设置 env USER=chen
(gdb) show env [name]    # 显示 env
(gdb) cd <dir>            # 切换 shell 目录
(gdb) pwd                 # 显示当前目录

(gdb) shell <command>    # 运行 shell 命令
```

## 断点(breakpoint)

```
SH(gdb) b/break linenum/func # 在 linenum 函数处断点
(gdb) b/break +/-offset      # 在 offset 处断点
(gdb) b/break filename:linenum # 在 filename 的 linenum 处断点
(gdb) b/break filename:func   # 在 filename 的 func 处断点
(gdb) b/break *address        # 在 address 处断点
(gdb) b/break                  # 在下一条指令处断点
(gdb) b/break if <condition>  # 在 condition 为真时断点

(gdb) info break [n]          # 显示断点 n 的信息
```

## SH/SHLIB(watchpoint)

```
SH# SHLIB
(gdb) watch <expr>          # SHLIB
(gdb) rwatch <expr>         # SHLIB
(gdb) awatch <expr>         # SHLIB

(gdb) info watchpoints      # SHLIB
```

## SH/SHLIB(catchpoint)

```
SH# SHLIBC++
(gdb) tcatch <event>        # SHLIB
(gdb) catch <event>         # SHLIB
# throw      SHLIBthrow
# catch      SHLIBcatch
# exec       SHLIBexecHP-UX
# fork       SHLIBforkHP-UX
# vfork      SHLIBvforkHP-UX
# load [file] SHLIBHP-UX
# unload [libname] SHLIBHP-UX
#
```

## SHLIB

```
SH# SHLIBgdbSHLIBdeleteSHLIBclearSHLIBdisableSHLIBenable
(gdb) clear          # SHLIB
(gdb) clear <fuction> # SHLIB
(gdb) clear <file:line> # SHLIB
(gdb) d/delete [n]/[m-n] # SHLIBm-n

# SHLIBdisableSHLIBdisableSHLIBGDBSHLIBenable
(gdb) disable [n]/[m-n] # SHLIBnSHLIBdisableSHLIBm-n

(gdb) enable [n]/[m-n] # SHLIBnSHLIBm-n
(gdb) enable once [n]/[m-n] # SHLIBnSHLIBdisableSHLIBm-n
(gdb) enable delete [n]/[m-n] # SHLIBSHLIBm-n
```

## SHLIB

```
SH# SHLIBbreakpointSHLIB
(gdb) condition <bnum> <expr> # SHLIBbnum
(gdb) condition <bnum>        # SHLIBbnum

# ignore SHLIB
(gdb) ignore <bnum> <count> # SHLIBhnumSHLIBcount
```



```
SH# [ ]command[ ]
(gdb) commands [bnum]
> ... commands list ...
> end          # [ ]bnum[ ]

[ ]
(gdb) break foo if x>0
(gdb) commands
> printf "x is %dn",x
> continue
> end
# [ ]foo[ ]x>0[ ]x[ ]foo[ ]0[GDB][ ]x[ ]
# [ ]commands[ ]end[ ]
```



```
SH# [ ]c++[ ]gdb[ ]
[ ]
(gdb) b String::after
[0] cancel
[1] all
[2] file:String.cc; line number:867
[3] file:String.cc; line number:860
[4] file:String.cc; line number:875
[5] file:String.cc; line number:853
[6] file:String.cc; line number:846
[7] file
> 2 4 6
Breakpoint 1 at 0xb26c: file String.cc, line 867.
Breakpoint 2 at 0xb344: file String.cc, line 875.
Breakpoint 3 at 0xafcc: file String.cc, line 846.
```



```
SH# [ ]c/continue[ ]step[ ]next[ ]
(gdb) c/continue [ignore-count]  # [ ]ignore-count[ ]

# [ ]debug[ ]VC[ ]stepin[ ]count[ ]count[ ]
(gdb) step <count>

# [ ]step-mode[ ]debug[ ]
(gdb) set step-mode on

# [ ]
(gdb) u/until

# [ ]stepi[ ]nexti[ ]"display/i $pc" [ ]
```

```
(gdb) si/stepi
(gdb) ni/stepi
```



```
SH# [REDACTED]UNIX[REDACTED]UNIX[REDACTED]SIGINT[REDACTED]Ctrl+C[REDACTED]S
# GDB[REDACTED]GDB[REDACTED]GDB[REDACTED]GDB[REDACTED]handle[REDACTED]
(gdb) handle <signal> <keywords...>
# [REDACTED]GDB[REDACTED]<signal>[REDACTED]SIG[REDACTED]SIG[REDACTED]SIGIO-SIGKILL[REDACTED]SIGIO[REDACTED]SIGKILL[REDACTED]SIGIO[
nostonp      # [REDACTED]GDB[REDACTED]
stop        # [REDACTED]GDB[REDACTED]
print       # [REDACTED]GDB[REDACTED]
noprnt      # [REDACTED]GDB[REDACTED]
pass/noignore # [REDACTED]GDB[REDACTED]GDB[REDACTED]
nopass/ignore # [REDACTED]GDB[REDACTED]

# [REDACTED]GDB[REDACTED]
(gdb) info signals
(gdb) info handle
```



```
SH# [REDACTED]singal[REDACTED]Ctrl+C[REDACTED]GDB[REDACTED]
(gdb) signal <singal>
# UNIX[REDACTED]1[REDACTED]15[REDACTED]<singal>[REDACTED]
# single[REDACTED]shell[REDACTED]kill[REDACTED]kill[REDACTED]GDB[REDACTED]single[REDACTED]
```



```
SH# [REDACTED]
(gdb) info threads      # [REDACTED]
(gdb) break <line> thread <threadno>      # [REDACTED]line[REDACTED]threadno[REDACTED]
(gdb) break <line> thread <threadno> if... # [REDACTED]line[REDACTED]threadno[REDACTED]

[REDACTED]
(gdb) break frik.c:13 thread 28 if bartab > lim
```



```
SH# [REDACTED]"[REDACTED]"Stack[REDACTED]GDB[REDACTED]
(gdb) bt/backtrace      # [REDACTED]
(gdb) bt/backtrace <n>  # [REDACTED]n[REDACTED]n[REDACTED]n[REDACTED]

# [REDACTED]
(gdb) f/frame <n>      # n[REDACTED]0[REDACTED]
(gdb) up <n>           # [REDACTED]n[REDACTED]n[REDACTED]
```

```
(gdb) down <n> # [ ]n[ ]n[ ]
```

```
# [ ]
```

```
(gdb) f/frame
```

```
# [ ]
```

```
(gdb) info f/frame
```

```
(gdb) info args # [ ]
```

```
(gdb) info locals # [ ]
```

```
(gdb) info catch # [ ]
```

## [[[

```
SH# [ ]-g[ ]
```

```
(gdb) l # [ ]
```

```
(gdb) l - # [ ]
```

```
(gdb) l + # [ ]
```

```
(gdb) l/list <linenum/func> # [ ]linenum[ ]function[ ]10
```

```
(gdb) l/list # [ ]list[ ]10
```

```
(gdb) l/list m,n # [ ]m[ ]n[ ]m[ ]n
```

```
(gdb) l/list -/+offset # [ ]offset
```

```
(gdb) l/list <file:line> # [ ]file[ ]line
```

```
(gdb) l/list <func> # [ ]func
```

```
(gdb) l/list <file:func> # [ ]file[ ]func
```

```
(gdb) l/list <*address> # [ ]address
```

```
(gdb) set listsize # [ ]
```

```
(gdb) show listsize # [ ]listsize
```

## [[[[

```
SH(gdb) forward-search <regexp> # [ ]regexp
```

```
(gdb) search <regexp> # [ ]
```

```
(gdb) reverse-search <regexp> # [ ]
```

## [[[[[

```
SH# [ ]-g[ ]GDB[ ]GDB[ ]
```

```
(gdb) dir/directory <dirname ... > # [ ]UNIX[ ]:"Windows[ ]";
```

```
(gdb) dir/directory # [ ]
```

```
(gdb) show directories # [ ]
```

## [[[[[

```
(gdb) disassemble func
```

--	--	--	--	--	--	--

\$22 = 0x65

--	--	--	--

```
(gdb) x/3uh 0x54320      # 0x5432000000000000h 3u
```



```
SH# GDB display
(gdb) display <expr>
(gdb) display/<fmt> <expr>
(gdb) display/<fmt> <addr>
# expr<fmt><addr> display GDB

# i s display
(gdb) display/i $pc
# $pc GDB/i
display GDB
(gdb) undisplay <dnums...>
(gdb) delete display <dnums...>
# dnums
# 2-5
(gdb) disable display <dnums...>
(gdb) enable display <dnums...>
# disable enable
(gdb) info display
# display GDB enable
```



```
SH# GDB
# 1 GDB
(gdb) set print address
(gdb) set print address on
(gdb) set print address off # 
(gdb) show print address # 

# 2
(gdb) set print array
(gdb) set print array on
(gdb) set print array off
(gdb) show print array
(gdb) show print elements # print elements
(gdb) set print elements <number-of-elements>
# <number-of-elements> GDB 0

# 3 off
(gdb) set print null-stop <on/off>

# 4 printf pretty GDB
(gdb) set print pretty on
(gdb) show print pretty # GDB

# 5 "nnn" nnn "65"
(gdb) set print sevenbit-strings <on/off>
(gdb) show print sevenbit-strings #
```



```
# 6
(gdb) set print union <on/off>
(gdb) show print union      # 

```

\$1 = {it = Tree, form = {tree = Acorn, bug = Cocoon}} #

\$1 = {it = Tree, form = {...}} #

```
# 7 C++ GDB GDB off
(gdb) set print object <on/off>
(gdb) show print object    # 

```

```
# 8 C++ on
(gdb) set print static-members <on/off>
(gdb) show print static-members # 

```

```
# 9 GDB
(gdb) set print vtbl <on/off>
(gdb) show print vtbl      # 

```

```
GDB print print GDB GDB $1, $2, $3 ..... print $1
```

```
SH# GDB GDB GDB set GDB UNIX $
(gdb) set $foo = *object_ptr
# GDB
(gdb) show convenience
# 
(gdb) set $i = 0
(gdb) print bar[$i++]->contents
# print bar[0]->contents, print bar[1]->contents
```

```
SH# ip sp print $ p $eip
(gdb) info registers      # ( )
(gdb) info all-registers  # 
(gdb) info registers regname# info rbp
```

```
SH# GDB GDB

#
```

```
# [REDACTED]GDB[REDACTED]GDB print[REDACTED]
(gdb) print x=4
# x=4[REDACTED]C/C++[REDACTED]x[REDACTED]4[REDACTED]Pascal[REDACTED]Pascal[REDACTED]x:=4[REDACTED]
# [REDACTED]GDB[REDACTED]
(gdb) whatis width
type = double
(gdb) p width
$4 = 13
(gdb) set width=47
Invalid syntax in expression.
# [REDACTED]set width[REDACTED]GDB[REDACTED]"Invalid syntax in expression"[REDACTED]set
# var[REDACTED]GDB[REDACTED]width[REDACTED]GDB[REDACTED]
(gdb) set var width=47
# [REDACTED]GDB[REDACTED]set var[REDACTED]GDB[REDACTED]
```

[REDACTED]

```
SH# [REDACTED]GDB[REDACTED]GDB[REDACTED]GDB jump[REDACTED]
(gdb) jump <linespec>
# [REDACTED]<linespec>[REDACTED]file:line[REDACTED]+num[REDACTED]
(gdb) jump <address>
# [REDACTED]<address>[REDACTED]jump[REDACTED]C
# [REDACTED]jump[REDACTED]"set $pc"[REDACTED]
(gdb) set $pc = 0x485
```

[REDACTED]

```
SH# [REDACTED]return[REDACTED]
(gdb) return
(gdb) return <expression>
# [REDACTED]return[REDACTED]<expression>[REDACTED]
```

[REDACTED]

```
SH(gdb) call <expr>
# [REDACTED]void[REDACTED]
# [REDACTED]—print[REDACTED]print[REDACTED]print[REDACTED]call[REDACTED]void[REDACTED]call[REDACTED]print[REDACTED]
```

## GDB [REDACTED]

```
SH(gdb) show language
# [REDACTED]GDB[REDACTED]C[REDACTED]
(gdb) info frame
# [REDACTED]
(gdb) info source
```

```
# GDBset language set languageGDB
(gdb) set language
The currently understood settings are:
local or auto Automatic setting based on source file
c Use the C language
c++ Use the C++ language
asm Use the Asm language
chill Use the Chill language
fortran Use the Fortran language
java Use the Java language
modula-2 Use the Modula-2 language
pascal Use the Pascal language
scheme Use the Scheme language
# set language
```

```
(gdb) q/quit # GDB
```