GNU Project Debugger (gdb)

Programa Exemplo

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

int tamanho(char *s)
{
  return strlen(s);
}

int main(int argc, char *argv[])
{
  char *a = "Teste";

  printf("Valor de a: %s\n", a);
  printf("Tamanho de a: %d\n", tamanho(a));

  for(int i = 0; i < 10; i++) {
    printf("%d\n", i);
  }
}
```

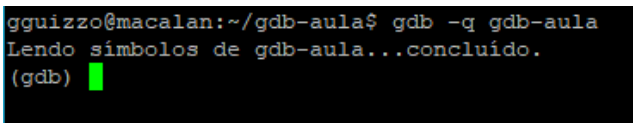
Compilando o Programa

Comando para compilação do programa, de modo que ele seja compatível com o gdb:

gcc -g -o gdb-aula gdb-aula.c

Executando o gdb

gdb -q gdb-aula



Comandos gdb

Utilizando o terminal do gdb, a depuração é feita por meio de comandos específicos. Alguns dos comandos que serão apresentados aqui:

run – Executa o programa;

```
(gdb) run
Starting program: /home/ppginf/gguizzo/gdb-aula/gdb-aula
Valor de a: Teste
Tamanho de a: 5
0
1
2
3
4
5
6
7
8
9
[Inferior 1 (process 25645) exited normally]
(gdb)
```

quit – Finaliza o gdb;

```
(gdb) quit
gguizzo@macalan:~/gdb-aula$
```

break – Adiciona um breakpoint;

```
(gdb) break main
Ponto de parada 1 at 0x40058f: file gdb-aula.c, line 12.
(gdb) break 7
Ponto de parada 2 at 0x400572: file gdb-aula.c, line 7.
(gdb)
```

watch – Adiciona um observador;

```
(gdb) watch i
Hardware watchpoint 3: i
(gdb) watch i>5
Hardware watchpoint 4: i>5
(gdb)
```

next – Avança uma instrução na depuração (não entra em métodos);

```
(gdb) run
Starting program: /home/ppginf/gguizzo/gdb-aula/gdb-aula

Breakpoint 1, main (argc=1, argv=0x7fffffffe5a8) at gdb-aula.c:12
12        char *a = "Teste";
(gdb) next
14        printf("Valor de a: %s\n", a);
(gdb) next
Valor de a: Teste
15        printf("Tamanho de a: %d\n", tamanho(a));
(gdb)
```

step – Avança uma instrução ou entra na chamada de um método;

```
15        printf("Tamanho de a: %d\n", tamanho(a));
(gdb) step

Breakpoint 2, tamanho (s=0x400684 "Teste") at gdb-aula.c:7
7         return strlen(s);
(gdb)
```

continue – Continua a execução do programa até o próximo breakpoint;

```
(gdb) continue
Continuando.
Tamanho de a: 5

Breakpoint 5, main (argc=1, argv=0x7fffffffe5a8) at gdb-aula.c:17
17        for(int i = 0; i < 10; i++){
(gdb)
```

finish – Finaliza a execução da função corrente;

```
(gdb) finish
Run till exit from #0  tamanho (s=0x400684 "Teste") at gdb-aula.c:7
0x00000000004005b9 in main (argc=1, argv=0x7fffffffe5a8) at gdb-aula.c:15
15        printf("Tamanho de a: %d\n", tamanho(a));
Value returned is $3 = 5
(gdb)
```

print – Imprime o valor de uma expressão;

```
(gdb) print i
$1 = 6
(gdb) print i*7
$2 = 42
(gdb)
```

display – Acompanha o valor de uma expressão a cada passo da depuração;

```
(gdb) display i
5: i = 4
(gdb) display i*i
6: i*i = 16
(gdb) next
18                printf("%d\n", i);
5: i = 5
6: i*i = 25
(gdb) next
5
17        for(int i = 0; i < 10; i++){
5: i = 5
6: i*i = 25
(gdb) next
18                printf("%d\n", i);
5: i = 6
6: i*i = 36
(gdb)
```

backtrace / where – Informação de stack;

```
(gdb) where
#0  tamanho (s=0x400684 "Teste") at gdb-aula.c:7
#1  0x00000000004005b9 in main (argc=1, argv=0x7fffffffe5a8) at gdb-aula.c:15
(gdb)
```

frame – Apresenta a instrução corrente;

```
(gdb) frame
#0  tamanho (s=0x400684 "Teste") at gdb-aula.c:7
7         return strlen(s);
(gdb)
```

info frame – Informações sobre a instrução corrente;

```
(gdb) info frame
Stack level 0, frame at 0x7fffffffe4a0:
 rip = 0x400572 in tamanho (gdb-aula.c:7); saved rip = 0x4005b9
 called by frame at 0x7fffffffe4d0
 source language c.
 Arglist at 0x7fffffffe490, args: s=0x400684 "Teste"
 Locals at 0x7fffffffe490, Previous frame's sp is 0x7fffffffe4a0
 Saved registers:
  rbp at 0x7fffffffe490, rip at 0x7fffffffe498
(gdb)
```

info args – Informações sobre os argumentos da função;

```
(gdb) info args
s = 0x400684 "Teste"
(gdb)
```

info locals – Informações sobre variáveis locais;

```
(gdb) info locals
i = 1
a = 0x400684 "Teste"
(gdb)
```

info breakpoints – Informações sobre breakpoints;

```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000400572 in tamanho at gdb-aula.c:7
        breakpoint already hit 1 time
2       breakpoint     keep y   0x000000000040058f in main at gdb-aula.c:12
        breakpoint already hit 1 time
3       breakpoint     keep y   0x00000000004005ca in main at gdb-aula.c:17
        breakpoint already hit 1 time
(gdb)
```

info line – Informações sobre a linha da instrução corrente;

```
(gdb) info line
Line 17 of "gdb-aula.c" starts at address 0x4005ca <main+74>
   and ends at 0x4005d3 <main+83>.
(gdb)
```

list – Mostra trechos do código fonte do programa;

```
10      int main(int argc, char *argv[])
11      {
12        char *a = "Teste";
13
14        printf("Valor de a: %s\n", a);
15        printf("Tamanho de a: %d\n", tamanho(a));
16
17        for(int i = 0; i < 10; i++){
18                printf("%d\n", i);
19        }
20      }
(gdb) list 17,19
17        for(int i = 0; i < 10; i++){
18                printf("%d\n", i);
19        }
(gdb)
```

clear – Apaga breakpoints ou observadores;

```
(gdb) clear main
Ponto de parada apagado 4
(gdb) clear tamanho
Ponto de parada apagado 5
(gdb) clear 17
Ponto de parada apagado 6
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)
```

kill – Mata a execução do programa;

```
(gdb) kill
Matar o programa que está sendo depurado? (y or n) y
(gdb)
```

set var – Define o valor para uma variável;

```
(gdb) info locals
a = 0x400684 "Teste"
(gdb) set var a="Modificado"
(gdb) info locals
a = 0x602010 "Modificado"
(gdb)
```

return – Retorna um valor para a função;

```
(gdb) step
tamanho (s=0x602010 "Modificado") at gdb-aula.c:7
7         return strlen(s);
(gdb) return 100
Make tamanho return now? (y or n) y
#0  0x00000000004005b9 in main (argc=1, argv=0x7fffffffe5a8) at gdb-aula.c:15
15        printf("Tamanho de a: %d\n", tamanho(a));
(gdb) next
Tamanho de a: 100
17        for(int i = 0; i < 10; i++){
(gdb)
```

CTRL + x a – Mostra ou esconde a janela de depuração;

```
   lqqgdb-aula.cqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
   x11       {                                                                x
B+ x12         char *a = "Teste";                                             x
   x13                                                                        x
   x14         printf("Valor de a: %s\n", a);                                 x
   x15         printf("Tamanho de a: %d\n", tamanho(a));                      x
   x16                                                                        x
B+>x17         for(int i = 0; i < 10; i++){                                   x
   x18                 printf("%d\n", i);                                     x
   x19         }                                                              x
   x20       }                                                                x
   x21                                                                        x
   x22                                                                        x
   x23                                                                        x
   mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
native process 27438 In: main                               L17   PC: 0x4005e7
```