

Depuração de código com o GDB

Antonio Marcos de Oliveira

Instituto Metrópole Digital - UFRN

22 de abril de 2018

Com a ajuda do depurador GDB, este trabalho tem o intuito de mostrar os valores das variáveis *arg1* e *arg2* após a chamada de cada função do código abaixo:

```
1      #include <iostream>
2
3      int funcX (int a, int b)
4      {
5          ++a;
6          b++;
7          int result = a + b;
8          return result;
9      }
10
11     int funcY (int* a, int b)
12     {
13         int* y = new int;
14         (*y) = (*a);
15         (*y) *= 5;
16         int result = (*y) + b;
17         return result;
18     }
19
20     void funcZ (int a, int b, int* result)
21     {
22         a++;
23         (*result) += a + 2*b;
24     }
25
26     int main(int argc, char* argv[])
27     {
28         int arg1 = 11;
29         int arg2 = 23;
30         funcX ( arg1, arg2);
31         funcY ( &arg1, arg2);
32         int resultado = 0;
33         funcZ (arg1, arg2, &resultado);
34
35         return 0;
36     }
```

Comandos do GDB utilizados:

Os comandos do GDB utilizados foram os seguintes:

- **break linha:** utilizado para adicionar um ponto de parada na linha especificada;
- **run:** inicia a execução do programa;
- **continue:** utilizado para dar continuidade ao programa até o próximo ponto de parada;
- **print var:** imprime o estado atual da variável especificada.

Depurando o código: Depois da compilação com a flag `-g` e `-O0` para que a depuração ocorresse sem problemas, a depuração foi iniciada. Antes de iniciar a execução do programa, foram adicionados 3 *breakpoints*, para cada chamada de função.¹

```

Activities Terminal
antonio@grower: ~/Documents/UFRN/LP1/UI/exercicios/lab02/gdb
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from gdb...done.
(gdb) break 31
Breakpoint 1 at 0x8d4: file gdb.cpp, line 31.
(gdb) break 32
Breakpoint 2 at 0x8e5: file gdb.cpp, line 32.
(gdb) break 34
Breakpoint 3 at 0x8ff: file gdb.cpp, line 34.
(gdb) run
Starting program: /home/antonio/Documents/UFRN/LP1/UI/exercicios/lab02/gdb/gdb

Breakpoint 1, main (argc=1, argv=0x7ffffffde78) at gdb.cpp:31
31      func1 ( &arg1, arg2);
(gdb) print arg1
$1 = 11
(gdb) print arg2
$2 = 23
(gdb) continue
Continuing.

Breakpoint 2, main (argc=1, argv=0x7ffffffde78) at gdb.cpp:32
32      int resultado = 0;
(gdb) print arg1
$3 = 11
(gdb) print arg2
$4 = 23
(gdb) continue
Continuing.

Breakpoint 3, main (argc=1, argv=0x7ffffffde78) at gdb.cpp:35
35      return 0;
(gdb) print arg1
$5 = 11
(gdb) print arg2
$6 = 23
(gdb) =

```

Figura 1: Inserção dos breakpoints à depuração

Na imagem acima está os passos da depuração, bem como o valor das variáveis *arg1* e *arg2* após a chamada de cada função.

¹O breakpoint utilizado foi o da linha ao invés do breakpoint da função devido ao fato que o o break da função é início da função, e não depois da execução. Logo, os breaks foram uma linha aos a chamada da função.