Grupo 3

Estático: Clang
Dinâmico: GDB

•••

Bárbara Varanda Rangel
Gabriel Martins de Miranda
Jadiel Teófilo Amorim de Oliveira
Marcos Cordeiro Fleury
Túlio de Carvalho Matias

CLang
Introdução e exemplos
...

Clang - definição

Clang é um front-end de um compilador para as linguagens C, C++, Objective-C e Objective-C++.

Tem como objetivo oferecer um substituto open source ao GCC, o conjunto de compiladores da GNU.

Clang atualmente implementa todo o padrão ISO C++ 1998 (incluindo os defeitos abordados na padrão ISO C++ 2003) e é considerado um compilador de qualidade de produção C++. Por padrão, Clang compila código C++ de acordo com o padrão C++ 98, com muitos recursos C++ 11 aceitos como extensões.

Clang - analisador estático

O analisador estático CLang é uma ferramenta de análise de código fonte para encontrar erros em C, C++ e Objective-C, sendo totalmente open source.

Tanto GCC quanto CLang tem a opção -fsyntax-only que faz o compilador apenas checar a sintaxe sem compilar os fontes.

Para chamar o analisar estático do CLang deve-se usar --analyse que tenta encontrar erros não encontrados com -fsyntax-only.

Para utilizá-lo: clang [clang-options] source_file.cpp

Exemplo 1: Divisão por zero

```
// divisão por zero
int main() {
  int z=0;
  int x = 1 / z; // warn

  return 0;
}
```

Com: clang-3.8 --analyze clang_test.c -o test

```
clang_test.c:25:7: warning: Value stored to 'x' during its initialization is never read
  int x = 1 / z; // warn
        ^   ~~~~~
clang_test.c:25:13: warning: Division by zero
  int x = 1 / z; // warn
            ~~^~~
2 warnings generated.
```

Exemplo 2: Expressão não inicializada

```
// expressão não inicializada
int main() {
  int x;

// warn: left expression
//is unitialized
  x |= 1;
  return 0;
}
```

Com: clang-3.8 --analyze clang_test.c -o test

clang_test.c:24:3: warning: Value stored to 'x' is never read

```
  x |= 1; // warn: left expression is unitialized
  ^    ~
```

clang_test.c:24:5: warning: The left expression of the compound assignment is an uninitialized value. The computed value will also be garbage

```
  x |= 1; // warn: left expression is unitialized
  ~ ^
```

2 warnings generated.

Exemplo 3: Verificação de ponteiros

```
namespace n6
{
 void f(int*) {}
 void useAfterDelete(int* p){
      delete p;

      //warn: use after free
      f(p);
 }
}
```

```
Com: clang++-3.8 --analyze clang_test.cc -o test
clang_test.cc:9:3: warning: Use of memory after it is freed
            f(p); //warn: use after free
            ^~~~
clang_test.cc:18:2: warning: Function call argument is an
uninitialized value
      testUseMiddleArgAfterDelete(value);
      ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

2 warnings generated.
```

Exemplo 4: Extrapolação de arrays

```
 int main()
{
        int vector[10];
        vector[12] = 2;

        return 0;
}
```

Com: clang-3.8 -fsyntax-only clang_test.c -o test
clang_test.c:6:2: warning: array index 12 is past the end of the array (which contains 10 elements) [-Warray-bounds]

```
                vector[12] = 2;
                ^      ~~
```

clang_test.c:5:2: note: array 'vector' declared here

```
                int vector[10];
                ^
```

1 warning generated.

Referências - Clang

"https://pt.wikipedia.org/wiki/Clang"
"https://llvm.org/svn/llvm-project/cfe/branches/release_33/www/cxx_status.html"
"http://clang-analyzer.llvm.org/"
"http://stackoverflow.com/questions/14072779/how-can-i-run-gcc-clang-for-static-analysis-warnings-only"
"https://github.com/scrooloose/syntastic/issues/512"
"http://clang-analyzer.llvm.org/available_checks.html"
"http://doc.qt.io/qtcreator/creator-clang-static-analyzer.html"

GDB
Introdução e exemplos
...

Analisador dinâmico - GDB

- GNU debugger
- Escrito em C
- Gratuito
- Multiplataforma
- Funciona com diversas linguagens (C, C++, Java, Fortran, etc.)
- Primeira versão - 1986
- Lançamento estável mais recente - 24 de fevereiro de 2016

Funcionalidades

- Funcionalidades básicas
  - Inicialização de programa especificando algo que possa afetar seu funcionamento.
  - Pausa do programa em condições especificadas.
  - Examinar o que aconteceu quando o seu programa parou de funcionar.
  - Realizar mudanças no seu programa para a realização de experimentos quanto ao bug encontrado.
- “Modo remoto”

Prós e contras

- Prós
  - Possibilidade de pontuar a localização de “segfalts” automaticamente.
  - Permite uma analise detalhada (manual) do programa em tempo de execução.
- Contras
  - Não possui verificação automática de “memory leaks”.

Documentação

- Documentação
  - <https://sourceware.org/gdb/onlinedocs/gdb/>
- Wiki
  - <https://sourceware.org/gdb/wiki/>

Exemplo - Um simples código a ser analisado

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

int expo(int valor);

int main(int argc, const char * argv[])
{
    int teste = expo(3);
    char *cara = 0;

    printf("%d\n", teste);
    teste = expo(-1);
    printf("%d\n", teste);
    expo(5);
    *cara = 3;
    return 0;
}
```

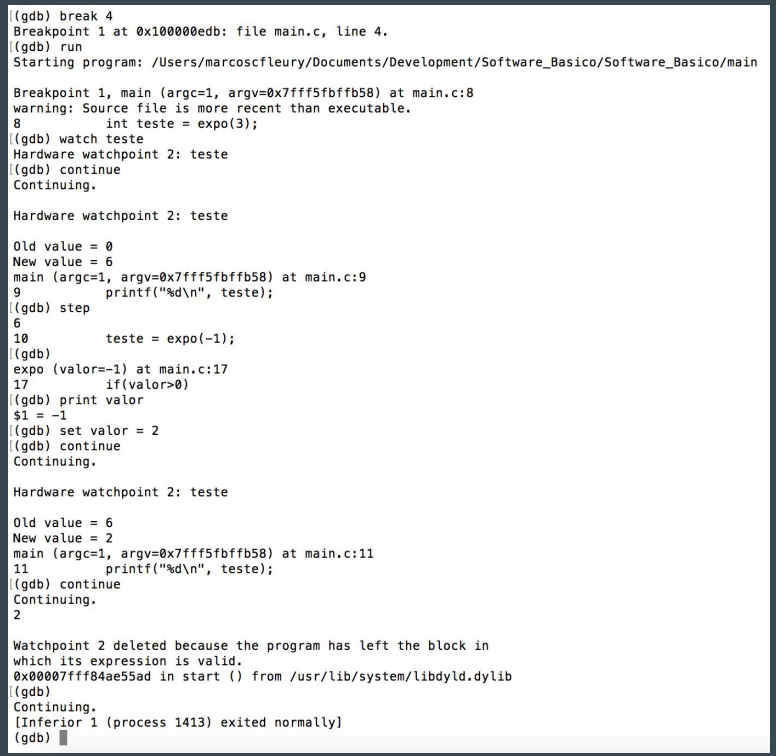
```
int expo(int valor){
  if(valor>0)
    return valor*expo(valor-1);
  return 1;
}
```

Análise - Código no GDB

O código em C já compilado é aberto no gdb:

```
GNU gdb (GDB) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin15.5.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from main...Reading symbols from /Users/marcoscfleury/Documents/Development/Software_Basico/Software_Basico/main.dSYM/Contents/Resources/DWARF/main...done.
done.
```

Análise - Breakpoints



```
(gdb) break 4
Breakpoint 1 at 0x100000edb: file main.c, line 4.
(gdb) run
Starting program: /Users/marcoscfleury/Documents/Development/Software_Basico/Software_Basico/main

Breakpoint 1, main (argc=1, argv=0x7fff5fbffb58) at main.c:8
warning: Source file is more recent than executable.
8           int teste = expo(3);
(gdb) watch teste
Hardware watchpoint 2: teste
(gdb) continue
Continuing.

Hardware watchpoint 2: teste

Old value = 0
New value = 6
main (argc=1, argv=0x7fff5fbffb58) at main.c:9
9           printf("%d\n", teste);
(gdb) step
6
10          teste = expo(-1);
(gdb)
expo (valor=-1) at main.c:17
17          if(valor>0)
(gdb) print valor
$1 = -1
(gdb) set valor = 2
(gdb) continue
Continuing.

Hardware watchpoint 2: teste

Old value = 6
New value = 2
main (argc=1, argv=0x7fff5fbffb58) at main.c:11
11          printf("%d\n", teste);
(gdb) continue
Continuing.
2

Watchpoint 2 deleted because the program has left the block in
which its expression is valid.
0x00007fff84ae55ad in start () from /usr/lib/system/libdyld.dylib
(gdb)
Continuing.
[Inferior 1 (process 1413) exited normally]
(gdb)
```

- Escolher onde interromper o código
- Observar as mudanças em uma variável
- Avançar por linha
- Verificar o estado de variáveis
- Modificar variáveis
- Analisar o código

Análise - Verificando erros

- Encontrar Segmentation Fault

```
(gdb) run
Starting program: /Users/marcoscfleury/Documents/Development/Software_Basico/Software_Basico/main
6
1

Program received signal SIGSEGV, Segmentation fault.
0x0000000100000f32 in main (argc=1, argv=0x7fff5fbffb58) at main.c:15
15          *cara = 3;
(gdb) where
#0  0x0000000100000f32 in main (argc=1, argv=0x7fff5fbffb58) at main.c:15
(gdb)
```

Referências - GDB

- Debugging with gdb. Disponível em <https://sourceware.org/gdb/onlinedocs/gdb/>. Acesso em 04 de setembro de 2016.
- GNU debugger. Disponível em <https://en.wikipedia.org/wiki/GNU_Debugger>. Acesso em 04 de setembro de 2016.
- "Palestra de Richard Stallman no Instituto Real de Tecnologia, Suiça(30-10-1986)"