Exercícios
Depuração de código C em Linux

Nesta aula pretende-se que os alunos façam *debugging* a um programa em C em Linux usando o *gdb (GNU Project Debugger* – https://www.gnu.org/software/gdb/*)*.

Exercício 1: utilização do *gdb*

Numa pasta de nome *lab3,* crie uma pasta de nome *ex1* e o ficheiro *main.c* com o conteúdo seguinte. Este programa é suposto calcular o factorial de um número, no entanto o resultado não está correcto. Compile e execute o programa para verificar.

```
1  #include <stdio.h>
2
3  long factorial(int num)
4  {
5        float result = 1;
6        while (--num) {
7              result *= num;
8        }
9        return result;
10 }
11
12 int main()
13 {
14       int num = 3;
15       long result = factorial(num);
16       printf("factorial(%d)=%ld\n", num, result);
17       return 0;
18 }
```

Para fazermos a depuração ao programa usando o *gdb* é necessário primeiro compilar usando a parâmetro *-g* para inserir os símbolos de depuração no executável.

```
$ gcc -g -o main main.c
```

Inicie-se o *gdb* dando como parâmetro o executável a depurar:

```
$ gdb main
Reading symbols from main...done.
(gdb)
```

Como sabemos que o problema deverá estar na função *factorial*, iremos colocar um *breakpoint* na linha 15 para o gdb parar a execução na invocação da função.

```
(gdb) break 15
Breakpoint 1 at 0x400578: file main.c, line 15.
```

Invoca-se então o comando *run* para executar o programa até ao breakpoint.

```
(gdb) run
Starting program: /home/jventura/Desktop/LabsSO/lab3/ex1/main
Breakpoint 1, main () at main.c:15
```

Chegando ao breakpoint, queremos que o *gdb* não continue o código no *main* mas que entre dentro da invocação da função *factorial*. Usa-se o comando *step*.

```
(gdb) step
factorial (num=3) at main.c:5
5           float result = 1;

(gdb) list
1     #include <stdio.h>
2
3     long factorial(int num)
4     {
5           float result = 1;
6           while (--num) {
7                 result *= num;
8           }
9           return result;
10    }
```

O comando *step* mostra-nos que estamos na linha 5 do programa. Em qualquer altura podemos usar o comando *list* para mostrar o conteúdo da função onde o *gdb* se encontra actualmente.

Visto que temos duas variáveis importantes para o cálculo do factorial, vamos obrigar o *gdb* a observar as variáveis de modo a controlarmos os valores que vão tomando ao longo do ciclo *for:*

```
(gdb) watch num
(gdb) watch result
```

Tendo os *watchpoints* definidos, e estando ainda na linha 5, podemos continuar o programa:

```
(gdb) continue
Hardware watchpoint 3: result

Old value = 0
New value = 1
factorial (num=3) at main.c:6
6           while (--num) {
```

Agora que estamos a observar as variáveis, podemos verificar que o código na linha #5 está a funcionar correctamente, colocando a variável *result* a 1. E chegamos então à linha #6. Para continuar, basta pressionarmos *Enter* que o *gdb* assume o último comando.

```
(gdb)
Hardware watchpoint 2: num
Old value = 3
New value = 2
0x0000000000400557 in factorial (num=2) at main.c:6
```

Aqui podemos já verificar que após a execução da linha #6, a variável *num* toma imediatamente o valor 2, sem que o valor 3 tenha sido usado anteriormente para calcular a multiplicação. Para provar que o valor 3 não é usado, continuamos.

```
(gdb)
Hardware watchpoint 3: result
Old value = 1
New value = 2
factorial (num=2) at main.c:6
```

Como verificamos, a variável *result* começa logo por multiplicar não pelo 3, mas sim pelo 2. Se continuarmos a execução até ao fim, iremos verificar que a variável result retornará 2 *1 invés do factorial 3 * 2 * 1.

Exercício 2: correcção do programa

Modifique o código da função *factorial* até corrigir o problema. Utilize o *gdb* com o *breakpoint* na invocação da função *factorial* de modo a verificar os vários passos do programa.

Exercício 3: cálculo de médias

O programa seguinte imprime a média de um array de inteiros, mas no entanto tem erros visto que a média é de 2,5. Use o *gdb* para encontrar o erro, e crie um *makefile* com as regras *build*, *debug* (compilar com parâmetro *-g*), *clean, run* e *rundebug* (que executa com *gdb*).

```
#include <stdio.h>

int main()
{
      int nums[] = {1, 2, 3, 4};
      int len = sizeof(nums);
      int sum;

      for (int i=0; i<len; i++) {
            sum += nums[i];
      }

      float avg = sum / len;
      printf("%f\n", avg);
      return 0;
}
```