

Fundamentos del Software: Ejercicio 2, Módulo II

Ejercicio 1

Utilizando los archivos disponibles en la carpeta de la sesión 8 (*main.cpp*, *factorial.cpp* y *hello.cpp*). Ejecuta la siguiente orden:

```
g++ main.cpp factorial.cpp hello.cpp -o ejecutable
```

Una vez compilado el código, utiliza la herramienta *gdb* para ver el código del programa principal. Describe qué sucede al intentar ver las líneas del código del *main* y cómo harías para ver el código por medio del depurador. Indica cómo se puede salir del depurador.

Una vez ejecutada la orden dada por el enunciado se crea el archivo *ejecutable*. Para ver las líneas del código del *main* utilizaremos la orden *list* del depurador *gdb*. Sin embargo, al intentar ejecutar dicha orden, se obtiene el siguiente resultado:

```
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ g++ main.cpp factorial.cpp hello.cpp -o ejecutable
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo símbolos desde ejecutable...(no se encontraron símbolos de depuración)hecho.
(gdb) list
No hay tabla de símbolos cargada. Use la orden «file».
(gdb) █
```

Es decir, no se muestran las líneas del código del *main* (que es lo que debería ocurrir). Para solucionarlo, primero eliminamos el archivo *ejecutable* creado anteriormente y vamos a ejecutar la siguiente orden para crear un nuevo archivo con el mismo nombre:

```
g++ -g main.cpp factorial.cpp hello.cpp -o ejecutable
```

Nótese que al comando dado por el enunciado le hemos añadido la opción *-g*. Si seguimos los mismos pasos que se muestran en la imagen anterior con esta nueva modificación el resultado es el siguiente:

```

juanma@juanma-VirtualBox:~/Escritorio/sesion08$ g++ -g main.cpp factorial.cpp hello.cpp -o ejecutable
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo simbolos desde ejecutable...hecho.
(gdb) list
1      #include <iostream>
2      #include "functions.h"
3
4      using namespace std;
5
6      int main(){
7          print_hello();
8          cout << endl;
9          cout << "The factorial of 7 is " << factorial(7) << endl;
10         return 0;
(gdb) █

```

Es decir, obtenemos el resultado que queríamos. Finalmente, para salir del depurador, utilizamos la orden *quit*:

```

juanma@juanma-VirtualBox:~/Escritorio/sesion08$ g++ -g main.cpp factorial.cpp hello.cpp -o ejecutable
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo simbolos desde ejecutable...hecho.
(gdb) list
1      #include <iostream>
2      #include "functions.h"
3
4      using namespace std;
5
6      int main(){
7          print_hello();
8          cout << endl;
9          cout << "The factorial of 7 is " << factorial(7) << endl;
10         return 0;
(gdb) quit
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ █

```

Ejercicio 2

¿Cómo harías para ejecutar la función *factorial* del ejercicio anterior? Muestra una captura de pantalla indicando el resultado y cómo has llegado a él.

Para ejecutar la función *factorial* del ejercicio anterior, mediante el depurador *gdb*, colocamos un punto de ruptura (*breakpoint*) en dicha función mediante el comando *break factorial*. Luego, utilizamos la orden *run* para ejecutar el programa hasta el punto de ruptura que hemos colocado en la función *factorial* (cabe destacar que en el *main* de nuestro programa se llama a la función *factorial(7)*, es decir, se quiere calcular el factorial del número 7). Finalmente, utilizamos la orden *next* (también se puede utilizar la orden *step*) para avanzar paso a paso en la ejecución de la función *factorial* (nótese que cuando termina de ejecutarse *factorial(7)*, si utilizamos la orden *next* nos devuelve la llave de cierre de la función *factorial*).

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejecutable...hecho.
(gdb) break factorial
Punto de interrupción 1 at 0x40094f: file factorial.cpp, line 4.
(gdb) run
Starting program: /home/juanma/Escritorio/sesion08/ejecutable
Hello World!

Breakpoint 1, factorial (n=7) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

Breakpoint 1, factorial (n=6) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

Breakpoint 1, factorial (n=5) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

Breakpoint 1, factorial (n=4) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

Breakpoint 1, factorial (n=3) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

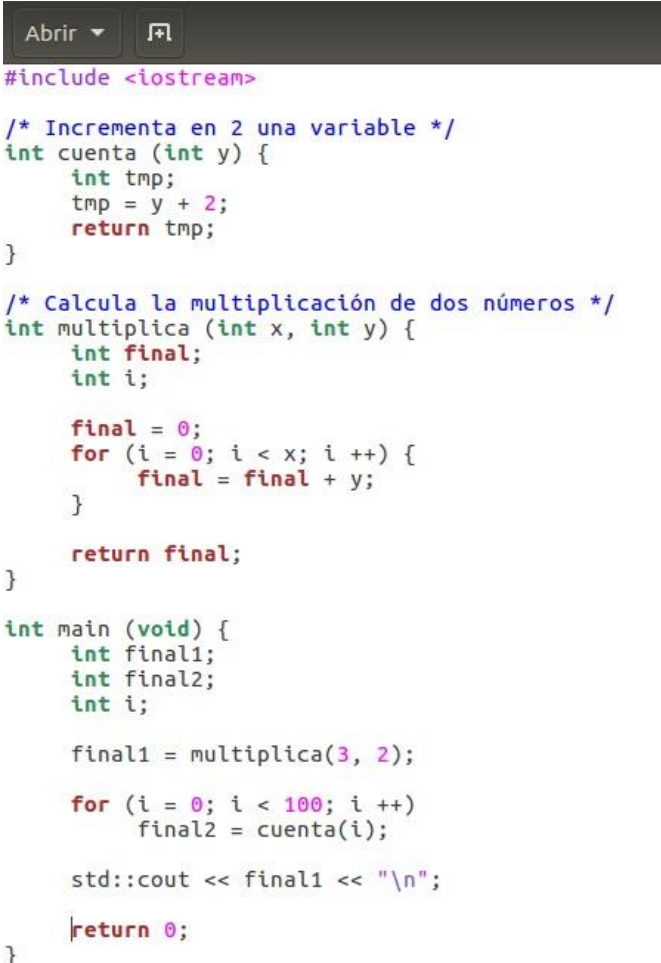
Breakpoint 1, factorial (n=2) at factorial.cpp:4
4      if(n!=1){
(gdb) next
5          return(n * factorial(n-1));
(gdb) next

Breakpoint 1, factorial (n=1) at factorial.cpp:4
4      if(n!=1){
(gdb) next
7      else return 1;
(gdb) next
8      }
(gdb) next
8      }
```

Ejercicio 3

Compila el código del siguiente programa y haz todas las configuraciones necesarias con gdb para mostrar el valor de la variable *final2* justo antes de que se le asigne ningún valor dentro del bucle *for*. Describe todos los pasos que has seguido e incluye una captura de pantalla de los mismos.

Primero copiamos el código dado por el enunciado en el editor de textos (en nuestro caso, hemos utilizado *gedit*) y lo guardamos con su correspondiente extensión *.cpp*. Le hemos dado al archivo el nombre *main2.cpp*.

A screenshot of a code editor window. At the top, there is a dark grey header bar with two buttons: 'Abrir' with a dropdown arrow and a file icon. Below the header, the C++ code is displayed with syntax highlighting. The code defines two functions: 'cuenta' which increments a value by 2, and 'multiplica' which calculates the product of two numbers using a for loop. The 'main' function calls 'multiplica(3, 2)' and then enters a for loop from 0 to 100, calling 'cuenta(i)' for each iteration. It prints the value of 'final1' and then returns 0.

```
#include <iostream>

/* Incrementa en 2 una variable */
int cuenta (int y) {
    int tmp;
    tmp = y + 2;
    return tmp;
}

/* Calcula la multiplicación de dos números */
int multiplica (int x, int y) {
    int final;
    int i;

    final = 0;
    for (i = 0; i < x; i++) {
        final = final + y;
    }

    return final;
}

int main (void) {
    int final1;
    int final2;
    int i;

    final1 = multiplica(3, 2);

    for (i = 0; i < 100; i++)
        final2 = cuenta(i);

    std::cout << final1 << "\n";

    return 0;
}
```

Luego, en la terminal de Linux, ejecutamos la siguiente orden:

```
g++ -g main2.cpp -o ejecutable2
```

De esta forma, obtendremos el archivo *ejecutable2*. Para ver las líneas del código del *main2* utilizaremos la orden *list 1,36* del depurador *gdb*.


```

juanma@juanma-VirtualBox:~/Escritorio/sesion08$ g++ -g main2.cpp -o ejecutable2
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable2
GNU gdb (Ubuntu 7.11.1-0ubuntu1-16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo símbolos desde ejecutable2...hecho.
(gdb) list 1,36
1      #include <iostream>
2
3      /* Incrementa en 2 una variable */
4      int cuenta (int y) {
5          int tmp;
6          tmp = y + 2;
7          return tmp;
8      }
9
10     /* Calcula la multiplicación de dos números */
11     int multiplica (int x, int y) {
12         int final;
13         int i;
14
15         final = 0;
16         for (i = 0; i < x; i++) {
17             final = final + y;
18         }
19
20         return final;
21     }
22
23     int main (void) {
24         int final1;
25         int final2;
26         int i;
27
28         final1 = multiplica(3, 2);
29
30         for (i = 0; i < 100; i++)
31             final2 = cuenta(i);
32
33         std::cout << final1 << "\n";
34
35         return 0;

```

Luego, colocamos un punto de ruptura en la línea 31 mediante la orden *break 31*. Elegimos la línea 31 ya que en dicha línea el bucle *for* no le ha asignado todavía ningún valor a la variable *final2*. Luego, utilizamos la orden *run* para ejecutar el programa hasta el punto de ruptura que hemos colocado. Finalmente, utilizamos la orden *print final2* para mostrar el valor de *final2* en dicho instante de la ejecución del programa. Nótese que el valor de dicha variable es 0, el cual era de esperar ya que todavía no se le ha asignado ningún valor dentro del bucle *for*.

```

35         return 0;
36     }
(gdb) break 31
Punto de interrupción 1 at 0x400823: file main2.cpp, line 31.
(gdb) run
Starting program: /home/juanma/Escritorio/sesion08/ejecutable2

Breakpoint 1, main () at main2.cpp:31
31             final2 = cuenta(i);
(gdb) print final2
$1 = 0
(gdb)

```

Ejercicio 4

Siguiendo el ejercicio anterior, haz todas las configuraciones necesarias utilizando el depurador *gdb* para obtener el valor de la variable *final2* cuando *i* vale 50. Muestra todos los comandos utilizados y el valor de las variables *final2* e *i*.

Comenzamos viendo las líneas del código del *main2* utilizaremos la orden *list 1,36* del depurador *gdb*.

```
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable2
GNU gdb (Ubuntu 7.11.1-0ubuntu1-16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo símbolos desde ejecutable2...hecho.
(gdb) list 1,36
1      #include <iostream>
2
3      /* Incrementa en 2 una variable */
4      int cuenta (int y) {
5          int tmp;
6          tmp = y + 2;
7          return tmp;
8      }
9
10     /* Calcula la multiplicación de dos números */
11     int multiplica (int x, int y) {
12         int final;
13         int i;
14
15         final = 0;
16         for (i = 0; i < x; i++) {
17             final = final + y;
18         }
19
20         return final;
21     }
22
23     int main (void) {
24         int final1;
25         int final2;
26         int i;
27
28         final1 = multiplica(3, 2);
29
30         for (i = 0; i < 100; i++)
31             final2 = cuenta(i);
32
33         std::cout << final1 << "\n";
34
35         return 0;
36     }
```

Después colocamos un punto de ruptura condicional en la línea 31 usando orden *break 31 if i == 50*. Dicha orden hará que el programa se detenga en dicho punto de ruptura cuando el valor de *i* sea igual a 50. Luego, utilizamos la orden *run* para ejecutar el programa hasta el punto de ruptura que hemos colocado. Finalmente, utilizamos las ordenes *print final2* y *print i* para mostrar el valor de ambas variables en ese instante de la ejecución del programa.

```
35         return 0;
36     }
(gdb) break 31 if i == 50
Punto de interrupción 1 at 0x400823: file main2.cpp, line 31.
(gdb) run
Starting program: /home/juanma/Escritorio/sesion08/ejecutable2

Breakpoint 1, main () at main2.cpp:31
31             final2 = cuenta(i);
(gdb) print final2
$1 = 51
(gdb) print i
$2 = 50
(gdb)
```

Ejercicio 5

Con el mismo código del ejercicio anterior no parece que la variable *final2* pueda ser mayor de 101. Utilizando el depurador *gdb* haz que, sin tocar la variable *final2*, esta variable tenga un valor por encima de 1000 al final del programa. Describe todos los pasos que has seguido para conseguirlo e incluye la captura de pantalla correspondiente.

Comenzamos viendo las líneas del código del *main2* utilizaremos la orden *list 1,36* del depurador *gdb*.

```
juanma@juanma-VirtualBox:~/Escritorio/sesion08$ gdb ejecutable2
GNU gdb (Ubuntu 7.11.1-0ubuntu1-16.5) 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-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<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"...
Leyendo símbolos desde ejecutable2...hecho.
(gdb) list 1,36
1      #include <iostream>
2
3      /* Incrementa en 2 una variable */
4      int cuenta (int y) {
5          int tmp;
6          tmp = y + 2;
7          return tmp;
8      }
9
10     /* Calcula la multiplicación de dos números */
11     int multiplica (int x, int y) {
12         int final;
13         int i;
14
15         final = 0;
16         for (i = 0; i < x; i++) {
17             final = final + y;
18         }
19
20         return final;
21     }
22
23     int main (void) {
24         int final1;
25         int final2;
26         int i;
27
28         final1 = multiplica(3, 2);
29
30         for (i = 0; i < 100; i++)
31             final2 = cuenta(i);
32
33         std::cout << final1 << "\n";
34
35         return 0;
36     }
```

Después colocamos un punto de ruptura condicional en la línea 7 usando orden *break 7 if y == 99*. Dicha orden hará que el programa se detenga en dicho punto de ruptura cuando el valor de entrada de la función *cuenta*, es decir, el valor de *y*, sea igual a 99, lo cual ocurre cuando la variable *i* del *for* tome su último valor posible, es decir, el valor 99 (decimos que es su último valor posible porque en el *for* observamos la expresión *i < 100*). Luego, utilizamos la orden *run* para ejecutar el programa hasta el punto de ruptura que hemos colocado. Mostramos el valor de la variable *tmp* mediante la orden *print tmp* y vemos que contiene el valor 101. Posteriormente, le asignamos a la variable *tmp* el valor 1001 mediante la orden *set variable tmp = 1001*. Finalmente, utilizamos la orden *next* para avanzar paso a paso en la ejecución del programa y justo antes de finalizar dicho programa lanzamos la orden *print final2* para mostrar el valor de dicha variable, pudiéndose observar que *final2 = 1001*.

```

35         return 0;
36     }
(gdb) break 7 if y == 99
Punto de interrupción 1 at 0x4007c6: file main2.cpp, line 7.
(gdb) run
Starting program: /home/juanma/Escritorio/sesion08/ejecutable2

Breakpoint 1, cuenta (y=99) at main2.cpp:7
7         return tmp;
(gdb) print tmp
$1 = 101
(gdb) set variable tmp = 1001
(gdb) print tmp
$2 = 1001
(gdb) next
8     }
(gdb) next
main () at main2.cpp:30
30         for (i = 0; i < 100; i ++){
(gdb) next
33             std::cout << final1 << "\n";
(gdb) print final2
$3 = 1001
(gdb) next
6
35         return 0;
(gdb) next
36     }
(gdb) next
__libc_start_main (main=0x4007fc <main()>, argc=1, argv=0x7fffffffdec8,
    init=<optimized out>, fini=<optimized out>, rtd_fini=<optimized out>,
    stack_end=0x7fffffffdeb8) at ../csu/libc-start.c:325
325     ../csu/libc-start.c: No existe el archivo o el directorio.
(gdb) █

```