



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

Adrian Ulises Mercado Martinez

*Profesor:*

Fundamentos de programación.

*Asignatura:*

10

*Grupo:*

10

*No de Práctica(s):*

*Integrante(s):*

González Bretón Arturo  
Paniagua Bautista Daniel  
Castañeda Mora Carlos

*No. de Equipo de  
cómputo empleado:*

49-50-51

14

*No. de Lista o Brigada:*

1

*Semestre:*

19/09/19

*Fecha de entrega:*

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

# Práctica10 Depuración de programas.

Castañeda Mora Carlos  
González Bretón Arturo  
Paniagua Bautista Daniel

17 octubre 2019

## 1 Introduction

Uno de los principales problemas al desarrollar aplicaciones son los errores de ejecución. Compilar un programa no es garantía suficiente de que funciona de la manera prevista. Es más, el ciclo de desarrollo de un programa está ocupado, en su mayoría por las tareas de diagnosticar y corregir los errores de ejecución. A los errores de ejecución en programas se les suele denominar en inglés bugs.

Hoy en día, los métodos que se utilizan para “depurar” los errores de un programa son múltiples y con diferentes niveles de eficacia. El método consistente en insertar llamadas a `printf` que escriben en pantalla mensajes es quizás el más ineficiente de todos ellos. En realidad lo que se precisa es una herramienta que permita ejecutar de forma controlada un programa, que permita suspender la ejecución en cualquier punto para poder realizar comprobaciones, ver el contenido de las variables, etc.

Esta herramienta se conoce con el nombre de depurador o, su término inglés, debugger. El depurador es un ejecutable cuya misión es permitir la ejecución controlada de un segundo ejecutable. Se comporta como un envoltorio dentro del cual se desarrolla una ejecución normal de un programa, pero a la vez permite realizar una serie de operaciones específicas para visualizar el entorno de ejecución en cualquier instante.

Más concretamente, el depurador permite:

- ejecutar un programa línea a línea
- detener la ejecución temporalmente en una línea de código concreta
- detener temporalmente la ejecución bajo determinadas condiciones
- visualizar el contenido de las variables en un determinado momento de la ejecución
- cambiar el valor del entorno de ejecución para poder ver el efecto de una corrección en el programa

Uno de los depuradores más utilizados en entornos Linux es `gdb` (Debugger de GNU).

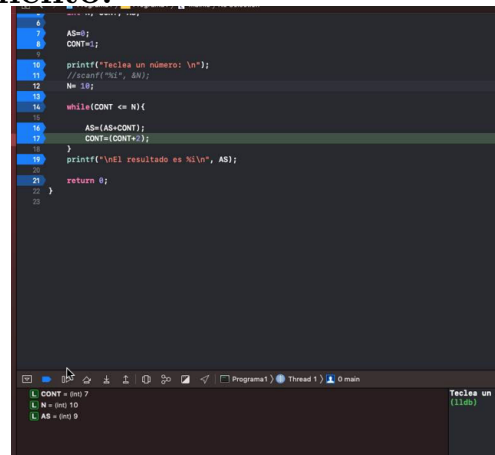
## 2 Desarrollo

### 1) Código del primer programa.



```
1 //
2 // p10ej1.c
3 // P10
4 //
5 // Created by Castaneda Mora Carlos on 10/17/19.
6 // Copyright © 2019 Castaneda Mora Carlos. All rights reserved.
7 //
8
9
10 #include <stdio.h>
11 void main() {
12     int N, CONT, AS;
13     AS=0;
14     CONT=1;
15     printf("TECLEA UN NUMERO: "); scanf("%i",&N); while(CONT<=N)
16     {
17         AS=(AS+CONT);
18         CONT=(CONT+2); }
19     printf("\nEL RESULTADO ES %i\n", AS); }
20
```

### 2) Se agregan breaks, para que cuando el programa llegue a ese punto pare su ejecución y regrese los valores de las variables que tienen en ese momento.



```
6 AS=0;
7 CONT=1;
8
9 printf("Tecllea un número: \n");
10 //scanf("%i",&N);
11 No se
12
13 while(CONT <= N){
14
15     AS=(AS+CONT);
16     CONT=(CONT+2);
17 }
18 printf("\nEl resultado es %i\n", AS);
19
20 return 0;
21 }
22
23
```

CONT = (int) 7  
N = (int) 10  
AS = (int) 9

Tecllea un número: (110)

### 3) Código del segundo programa.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5
6     int *apuntador = NULL;
7     apuntador = (int *) malloc(sizeof(int));
8     *apuntador = 4;
9     printf("%d, %p\n", *apuntador, apuntador);
10    free(apuntador);
11    return 0;
12 }
13
```

4, 0x103002960  
Program ended with exit code: 0

### 4) Se le argegan break al nuevo código para comprobar el valor de las variables.

```
1 #include <stdio.h>
2
3 int main(){
4
5     int *apuntador = NULL;
6
7     *apuntador = 4;
8
9     return 0;
10 }
11
```

Thread 1: breakpoint 1.1

Program1 - 3429 Thread 1 0 main  
apuntador = (int \*) 0x7f9a0f5508 (lldb)

### 5) Para depurar el programa con gdb, comando de Linux, ongresamos a una maquina virtual creada en el servidor para cada usuario, aquí vemos como entramos al servidor.

```
p11 — fp15alu11@samba:~ — ssh fp15alu11@192.168.3.200 — 80x24
The authenticity of host '192.168.3.200 (192.168.3.200)' can't be established.
RSA key fingerprint is SHA256:jTgFsbvP7IaIpwchV27DaUa9i2pvAVVZwZzbIneOF8.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '192.168.3.200' (RSA) to the list of known hosts.
fp15alu11@192.168.3.200's password:
Permission denied, please try again.
fp15alu11@192.168.3.200's password:

  Samba

-bash: aviso: setlocale: LC_CTYPE: no se puede cambiar el local (UTF-8)
[fp15alu11@samba ~]$ ls
Escritorio
[fp15alu11@samba ~]$ ls
Escritorio  p11_e1.c
[fp15alu11@samba ~]$
```

6)Adentro, compilamos el programa y ejecutamos.

```
p11 — fp15alu11@samba:~ — ssh fp15alu11@192.168.3.200 — 80x24
/usr/lib/gcc/x86_64-redhat-linux/4.7.2/../../../../lib64/crti.o: (.init+0x0): first
defined here
/usr/lib/gcc/x86_64-redhat-linux/4.7.2/crtend.o: (.tm_clone_table+0x0): multiple
definition of `__TMC_END__'
/bin/ld: error in p11_e1(.eh_frame); no .eh_frame_hdr table will be created.
collect2: error: ld devolvi? el estado de salida 1
[fp15alu11@samba ~]$ gdb ./p11_e1
-bash: gdb./p11_e1: No existe el fichero o el directorio
[fp15alu11@samba ~]$ ls
Escritorio  p11_e1  p11_e1.c
[fp15alu11@samba ~]$ quit
bash: quit: command not found...

[fp15alu11@samba ~]$
[fp15alu11@samba ~]$ gcc -g p11_e1.c -o p11_e1
[fp15alu11@samba ~]$ ./p11_e1
Lista
Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7
[fp15alu11@samba ~]$
```

7)Utilizamos el comando gdb para depurar el prgrama y agregar breaks.

```
[MMA@samba ~]$ gdb ./pr1
GNU gdb (GDB) Fedora (7.4.50.20120120-42.fc17)
Copyright (C) 2012 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /users/profesor/MMA/pr1...done.
(gdb) l 1-15
Function "1-15" not defined.
(gdb) list 1-15
Function "1-15" not defined.
(gdb) l
1      #include <stdio.h>
2      #define TAM 5
3      int main()
4      {
5          int lista[TAM] = {10, 8, 5, 8, 7};
6
7          int i = 0 ;
8
9          printf("\tlista\n");
10         while(i < 5)
(gdb) l
11         {
12             printf("\nCalificación del alumno %d es %d",i,lista[i]);
13             i++;
14         }
15         printf("\n");
16         return 0;
17     }
(gdb)

Function "1-15" not defined.
(gdb) list 1-15
Function "1-15" not defined.
(gdb) l
1      #include <stdio.h>
2      #define TAM 5
3      int main()
4      {
5          int lista[TAM] = {10, 8, 5, 8, 7};
6
7          int i = 0 ;
8
9          printf("\tlista\n");
10         while(i < 5)
(gdb) l
11         {
12             printf("\nCalificación del alumno %d es %d",i,lista[i]);
13             i++;
14         }
15         printf("\n");
16         return 0;
17     }
(gdb) b 9
Breakpoint 1 at 0x40059e: file programa1.c, line 9.
(gdb) b 10
Breakpoint 2 at 0x4005a8: file programa1.c, line 10.
(gdb) b 13
Breakpoint 3 at 0x4005c7: file programa1.c, line 13.
(gdb) b 15
Breakpoint 4 at 0x4005d1: file programa1.c, line 15.
(gdb) b 16
Breakpoint 5 at 0x4005db: file programa1.c, line 16.
(gdb) d 16
No breakpoint number 16.
```

### 3 Conclusión

El depurador es una herramienta sumamente útil, no sólo cuando el código trabaja con apuntadores, pues en muchas ocasiones se pueden presentar errores en la lógica de programación (el compilador no detecta estos errores). Desafortunadamente no pudimos utilizar el gdb, ya que no funciona en MacOS, sin embargo, pudimos depurar varios programas con ayuda de XCode (un IDE propio de MacOS); considerando lo anterior, los objetivos de la práctica se cumplieron en su totalidad. Arturo

Un depurador es una herramienta muy importante en la vida de un programador, ya que hay ocasiones en la que el compilador no detecta los errores, por eso al agregar puntos donde el programa se detiene y envía el valor de las variables, ayuda a detectar eficientemente los problemas en el código o la parte que no funciona. A pesar de haber probado el comando gdb con el tiempo encima, logramos entender el funcionamiento de este y su función en la corrección de programas desde el terminal. Carlos.

### 4 Bibliografía

<https://es.wikipedia.org/wiki/Depurador>  
[http://www.it.uc3m.es/pbasanta/asng/course\\_notes/debugger.es.html](http://www.it.uc3m.es/pbasanta/asng/course_notes/debugger.es.html)