



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* MC. Alejandro Esteban Pimentel Alarcón

*Asignatura:* Fundamentos de Programación

*Grupo:* 3

*No de Práctica(s):* Práctica 10

*Integrante(s):* Martínez Marcelino Dalila

*No. de Equipo de  
cómputo empleado:* No. de cuenta: 313080119

*No. de Lista o Brigada:*

*Semestre:* 2020-1

*Fecha de entrega:* Lunes 21 de octubre de 2019

*Observaciones:* Muy bien

**CALIFICACIÓN:** 10

## Práctica No. 10

### Introducción:

En esta práctica el estudiante aprenderá técnicas básicas de depuración en programas en lenguaje C haciendo uso de gdb (GDB- Gnu Project Debugger), el cual es un depurador estándar para GNU, una herramienta que permitirá al estudiante correr el programa y dará la posibilidad de detenerlo cuando se cumple la condición del programa, además permitirá avanzar paso a paso el programa lo que facilitara al estudiante poder localizar y revisar errores de manera precisa el flujo de ejecución del programa, y de esta manera poder corregir los posibles errores que se lleguen a encontrar.

### Objetivo:

Aprender las técnicas básicas de depuración de programas en C para revisar de manera precisa el flujo de ejecución de un programa y el valor de las variables; en su caso, corregir posibles errores.

### Ejemplo 1.

Conoceremos y aprenderemos a utilizar los comandos de gdb, utilizando el programa ejemplo1.c como programa prueba.

🚦 El programa a ejecutar es ejemplo1.c, y es el siguiente:



```
C:\cygwin64\home\DELL\programa\ejemplo1.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

ejemplo1.c
1 #include <stdio.h>
2
3 int main(int argc, char * argv[]) {
4
5     // Asignamos variables
6     int numero = 10;
7     int lista[numero];
8     char caracter = 'B';
9     float numeroReal = 89.8;
10    long int suma = 0;
11    double promedio;
12
13    // Mostramos texto y valores
14    printf("Primero texto solo\n");
15    printf("Luego podemos poner un entero: %i\n", numero);
16    printf("También podemos poner un caracter: %c\n", caracter);
17    printf("Un numero real: %.2f\n", numeroReal);
18
19    // Podemos llenar la lista con valores
20    for(int i = numero ; i >= numero ; i++){
21        lista[i] = i;
22    }
23
24    // Y ahora podemos hacer calculos con la lista
25    for(int i = numero ; i >= numero ; i++){
26        suma += lista[i];
27    }
28    promedio = suma / numero;
29    printf("La suma es: %li\n", suma);
30    printf("El promedio es: %lf\n", promedio);
31
32    return 0;
33 }
```

- Primero haremos una compilación como lo hemos hecho ya con anterioridad.

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gcc ejemplo1.c -o ejemplo1

DELL@DESKTOP-N3V5QTU ~/progra
$
```

- Vemos que hace la compilación sin ningún problema, por lo que podemos decir que al parecer no tiene ningún error y podremos ejecutar el programa.
- Al ejecutar el programa vemos que hace lo siguiente:

```
DELL@DESKTOP-N3V5QTU ~/progra
$ ./ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80
Segmentation fault ('core' generado)
```

- Sin embargo, podemos ver que en la última línea del programa ejecutado aparece un mensaje que dice “Segmentation fault ('core' generado)”, que significa violación de segmento.

```
Segmentation fault ('core' generado)
```

- Para ver el porqué de esta línea usaremos la herramienta del depurador gdb. Para ello escribiremos lo siguiente:

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gcc -g ejemplo1.c -o ejemplo1
```

- Es necesario incluir la '-g' bandera para que el compilador incluya la información referente a los símbolos y el código fuente en el archivo objeto del programa. Después esta información es incluida en el programa ejecutable
- Después lo compilaremos utilizando gdb.

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gdb ./ejemplo1
```

- Y al darle enter aparecerá lo siguiente, lo que indica que ya hemos entrado al depurador.

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gdb ./ejemplo1
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ./ejemplo1...done.
(gdb)
```

- Escribiremos run y el depurador empezará a correr el programa e indicará donde se encuentra el error.

```
Reading symbols from ./ejemplo1...done.
(gdb) run
Starting program: /home/DELL/progra/ejemplo1
[New Thread 2096.0xc38]
[New Thread 2096.0xdc4]
[New Thread 2096.0x4fc]
[New Thread 2096.0x1300]
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

Thread 1 "ejemplo1" received signal SIGSEGV, Segmentation fault.
0x000000010040118b in main (argc=1, argv=0xffffcc30) at ejemplo1.c:21
21             lista[i] = i;
(gdb) |
```

- La violación de segmento se encuentra en la línea 21.

```
Thread 1 "ejemplo1" received signal SIGSEGV, Segmentation fault.
0x000000010040118b in main (argc=1, argv=0xffffcc30) at ejemplo1.c:21
21             lista[i] = i;
(gdb) |
```

- El paso siguiente será escribir list o l y aparecerá lo siguiente:

```
(gdb) list
16             printf("También podemos poner un caracter: %c\n", caracter);
17             printf("Un numero real: %.2f\n", numeroReal);
18
19             // Podemos llenar la lista con valores
20             for(int i = numero ; i >= numero ; i++){
21                 lista[i] = i;
22             }
23
24             // Y ahora podemos hacer calculos con la lista
25             for(int i = numero ; i >= numero ; i++){
(gdb) |
```

- Para terminar la ejecución del programa y poder salir ponemos quit o q, y aparecerá el siguiente mensaje, preguntándonos si deseamos salir si o no. Al ponerle y (yes) hemos salido.

```
(gdb) quit
A debugging session is active.

        Inferior 1 [process 2096] will be killed.

Quit anyway? (y or n)
```

```
Quit anyway? (y or n) y
DELL@DESKTOP-NJV5QTU ~/progra
$ |
```

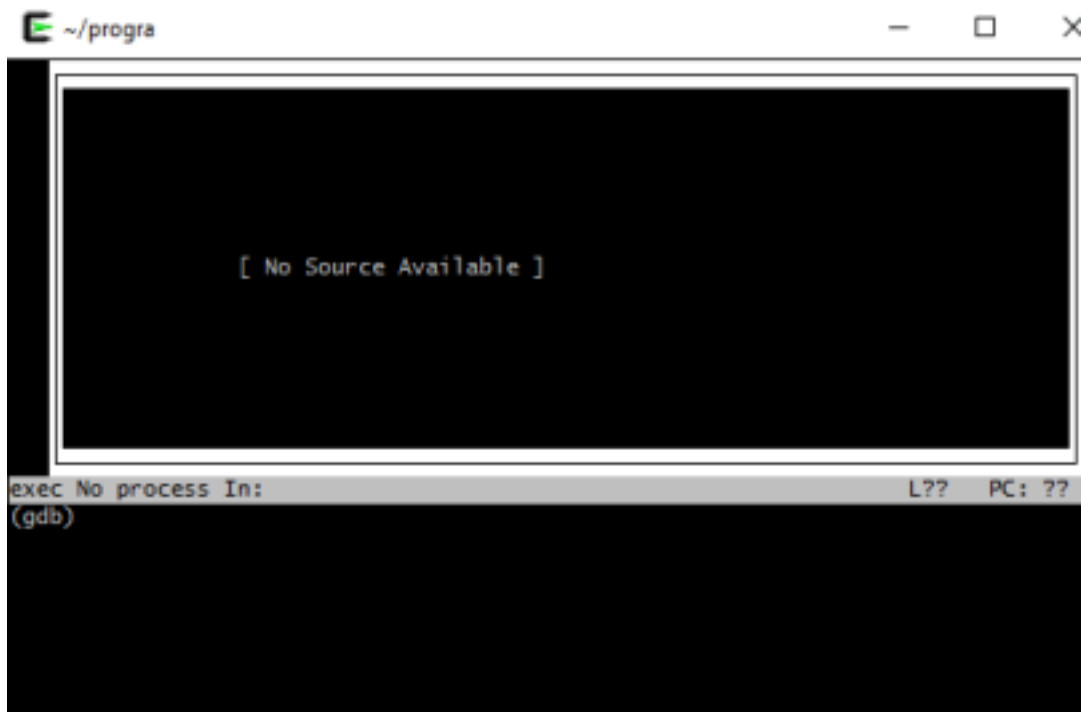
- Ahora llamaremos al programa ejemplo1 para que se ejecute bajo gdb, para ello escribiremos lo siguiente:

```

DELL@DESKTOP-NJVSQTU ~/progra
$ gdb ejemplo1
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ejemplo1...done.
(gdb) |

```

Utilizando el comando CTRL+x+a lo cual abra una ventanilla como la siguiente.



Escribiremos start lo que hará que el programa comience y se vea en la ventanilla.

```

exec No process in: L?? PC: ??
(gdb) start

```

```
~/progra
ejemplo1.c
8+> 3 int main(int argc, char * argv[]) {
4
5 // Asignamos variables
6 int numero = 10;
7 int lista[numero];
8 char caracter = '8';
9 float numeroReal = 89.8;
10 long int suma = 0;
11 double promedio;
12
13 // Mostramos texto y valores
14 printf("Primero texto solo\n");
15 printf("Luego podemos poner un entero: %i\n", numero);

native Thread 7776.0xabc In: main L3 PC: 0x1004010a0
[New Thread 7776.0xabc]
[New Thread 7776.0x22f4]
[New Thread 7776.0x23c4]
[New Thread 7776.0x1d98]

Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001004010a0 in main (
  argc=1, argv=0xffffcc30) at ejemplo1.c:3
(gdb)
```

Utilizamos el comando next o n para movernos en las líneas dentro del programa.

```
~/progra
ejemplo1.c
3 int main(int argc, char * argv[]) {
4
5 // Asignamos variables
6 int numero = 10;
7 int lista[numero];
8 char caracter = '8';
9 float numeroReal = 89.8;
10 long int suma = 0;
11 double promedio;
12
13 // Mostramos texto y valores
14 printf("Primero texto solo\n");
15 printf("Luego podemos poner un entero: %i\n", numero);

native Thread 7776.0xabc In: main L6 PC: 0x1004010a6
[New Thread 7776.0x23c4]
[New Thread 7776.0x1d98]

Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001004010a0 in main (
  argc=1, argv=0xffffcc30) at ejemplo1.c:3
(gdb) next
[New Thread 7776.0xee4]
(gdb) |
```

Sin volver a escribir n y solo dándole enter podremos movernos de línea en línea.

```
~/progra
ejemplo1.c
3      int main(int argc, char * argv[]) {
4
5          // Asignamos variables
6          int numero = 10;
7      int lista[numero];
8          char caracter = 'B';
9          float numeroReal = 89.8;
10         long int suma = 0;
11         double promedio;
12
13         // Mostramos texto y valores
14         printf("Primero texto solo\n");
15         printf("Luego podemos poner un entero: %i\n", numero);
16
17         // Mostramos texto y valores
18         printf("También podemos poner un caracter: %c\n", caracter);
19         printf("Un numero real: %.2f\n", numeroReal);
20         lista[i] = i;
21     }
22 }
```

native Thread 7776.0xabc In: main L7 PC: 0x1004010ad  
[New Thread 7776.0x1d98]  
Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001004010a0 in main (argc=1, argv=0xffffcc30) at ejemplo1.c:3  
(gdb) next  
[New Thread 7776.0xee4]  
(gdb) n  
(gdb) |

Al hacer estos movimientos en algún momento la pantalla de nuestra ventanilla aparecerá de la siguiente forma:

```
~/progra
11         double promedio;
12
13         // Mostramos texto y valores
14         printf("Primero texto solo\n");
14         printf("Primero texto solo\n");entero: %i\n", numero);
15         printf("Luego podemos poner un entero: %i\n", numero);cter
15         printf("Luego podemos poner un entero: %i\n", numero);
16         printf("También podemos poner un caracter: %c\n", caracter
16         printf("También podemos poner un caracter: %c\n", caracter
17         printf("Un numero real: %.2f\n", numeroReal);
21         lista[i] = i;
22     }
23 }
```

native Thread 7776.0xabc In: main L14 PC: 0x10040111e  
Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001006010a0 in main (argc=1, argv=0xffffcc30) at ejemplo1.c:3  
(gdb) next  
[New Thread 7776.0xee4]  
(gdb) n  
[New Thread 7776.0xe20]  
Primero texto solo  
Luego podemos poner un entero: 10  
También podemos poner un caracter: B  
(gdb) |

Para que vuelva a su estado normal ponemos CTRL+I.



```

~/progra
ejemplo1.c
9      float numeroReal = 89.8;
10     long int suma = 0;
11     double promedio;
12
13     // Mostramos texto y valores
14     printf("Primero texto solo\n");
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
> 17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;

```

native Thread 7776.0xabc In: main L17 PC: 0x10040114d

Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001004010a0 in main (argc=1, argv=0xffffcc30) at ejemplo1.c:3

(gdb) next

[New Thread 7776.0xee4]

(gdb) n

[New Thread 7776.0xe20]

(gdb)

- Al llegar a la línea que nos indicaba que había una violación de segmento al darle next este ahora solo se mueve entre la línea 21 y la línea anterior.

```

~/progra
9      float numeroReal = 89.8;
16     printf("También podemos poner un caracter: %c\n", caracter);
> 17     printf("Un numero real: %.2f\n", numeroReal);
17     printf("Un numero real: %.2f\n", numeroReal);
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
20     for(int i = numero ; i >= numero ; i++){
> 21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }

```

native Thread 7776.0xabc In: main L17 PC: 0x10040114d

Thread 1 "ejemplo1" hit Temporary breakpoint 1, 0x00000001004010a0 in main (argc=1, argv=0xffffcc30) at ejemplo1.c:3

(gdb) next

[New Thread 7776.0xee4]

(gdb) n

[New Thread 7776.0xe20]

(gdb) n

Un numero real: 89.80

(gdb)

- Si le seguimos dando enter en algún momento aparecerá que el programa no está corriendo.

```

native No process In: L?? PC: ??
[Thread 4652.0x18e0 exited with code 35584]
[Thread 4652.0x18ec exited with code 35584]
[Thread 4652.0x6e4 exited with code 35584]
[Thread 4652.0x1938 exited with code 35584]
[Inferior 1 (process 4652) exited with code 0105400]
The program is not being run.
The program is not being run.
(gdb) |

```

- Terminamos la ejecución del programa.



```
~/progra
ejemplo1.c
3 int main(int argc, char *argv[]) {
4
5     // Asignamos variables
6     int numero = 10;
7     int lista[numero];
8     char caracter = 'B';
9     float numeroReal = 89.8;
10    long int suma = 0;
11    double promedio;
12
13    // Mostramos texto y valores
14    printf("Primero texto solo\n");
15    printf("Luego podemos poner un entero: %i\n", numero);
16
17    return 0;
18 }
```

```
native No process in: L?? PC: ??
[Thread 4652.0x18e0 exited with code 35584]
[Thread 4652.0x18ec exited with code 35584]
[Thread 4652.0x6e4 exited with code 35584]
[Thread 4652.0x1938 exited with code 35584]
[Inferior 1 (process 4652) exited with code 0105400]
The program is not being run.
The program is not being run.
(gdb) q
```

- Volvemos a llamar al programa bajo gdb y escribimos break 20 para poner un punto de interrupción en la línea 20 del programa.

```
~/progra
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 ejemplo1...done.
(gdb)
DELL@DESKTOP-NJVSQTU ~/progra
$ gdb ejemplo1
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ejemplo1...done.
(gdb) break 20
```

- Le damos enter y nos indica que hay un punto de interrupción en la línea 20.

```
(gdb) break 20
Breakpoint 1 at 0x100401176: file ejemplo1.c, line 20.
(gdb) |
```

- Ponemos run y corremos nuevamente el programa. El programa se ejecuta y se detiene en la línea 20.

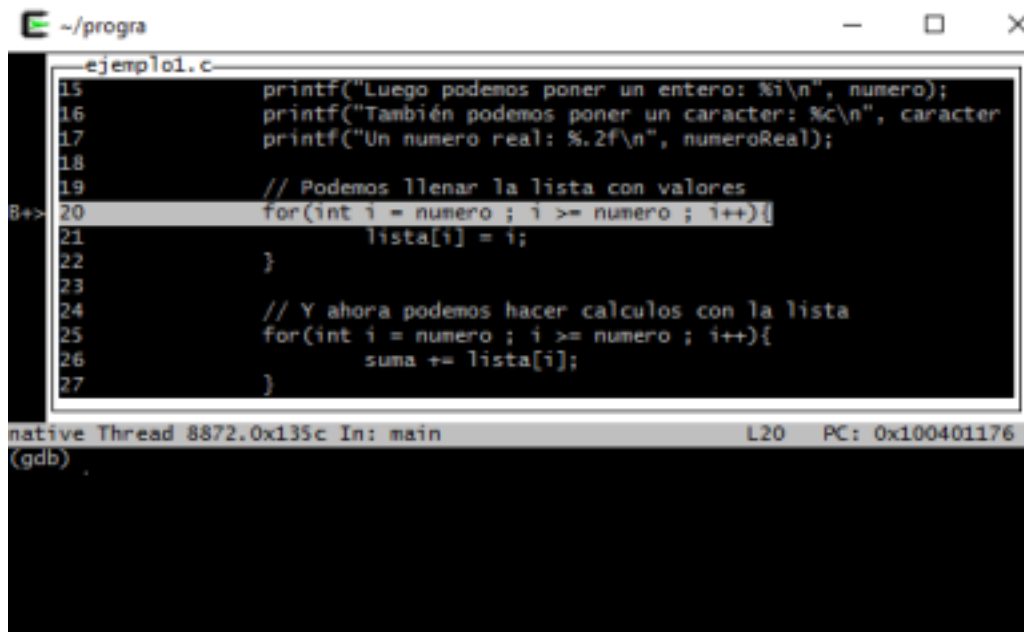
```

(gdb) run
Starting program: /home/DELL/progra/ejemplo1
[New Thread 8872.0x135c]
[New Thread 8872.0x221c]
[New Thread 8872.0x19fc]
[New Thread 8872.0x16c0]
Primer texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

Thread 1 "ejemplo1" hit Breakpoint 1, main (argc=1, argv=0xffffcc30)
at ejemplo1.c:20
20         for(int i = numero ; i >= numero ; i++){
(gdb)

```

- Entramos nuevamente a la ventanilla con CTRL+x+a y vemos que en la línea 20 aparece del lado izquierdo una B+.

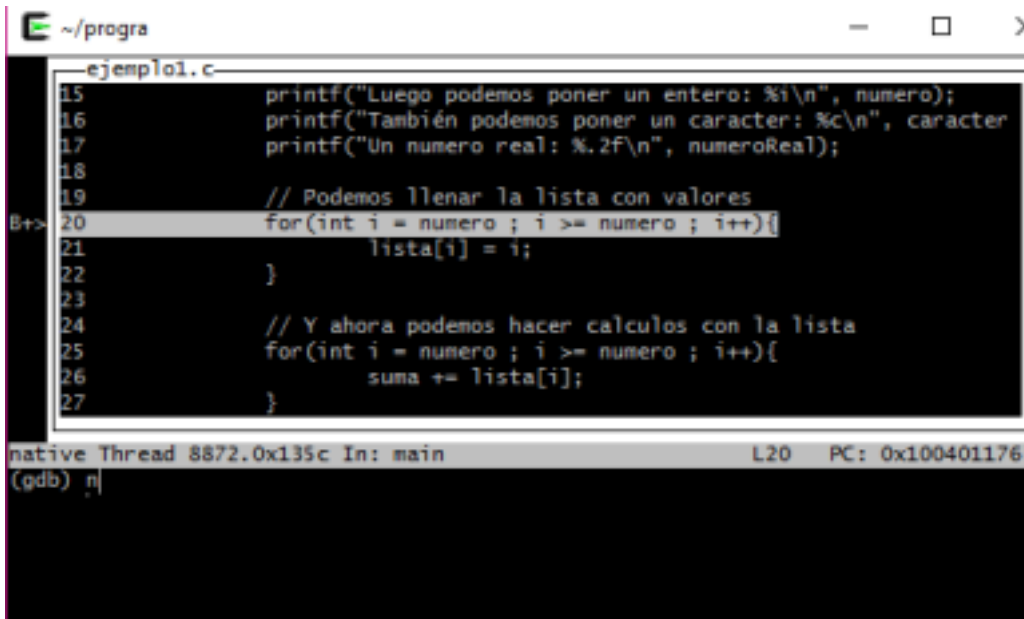


```

~/progra
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }
native Thread 8872.0x135c In: main L20 PC: 0x100401176
(gdb)

```

- Le damos next para movernos una línea abajo.

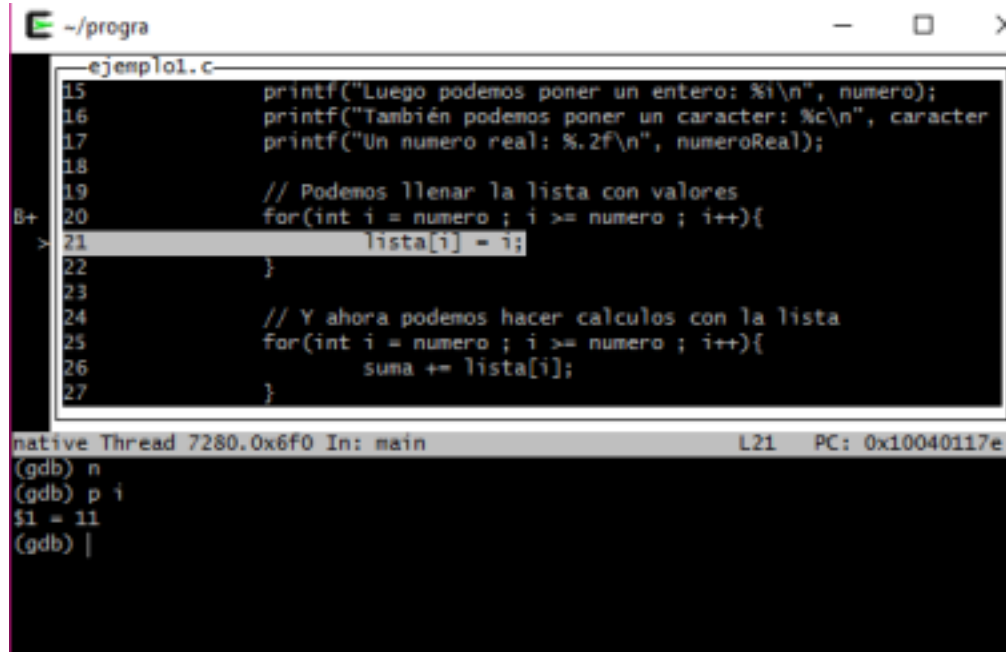


```

~/progra
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }
native Thread 8872.0x135c In: main L20 PC: 0x100401176
(gdb) n

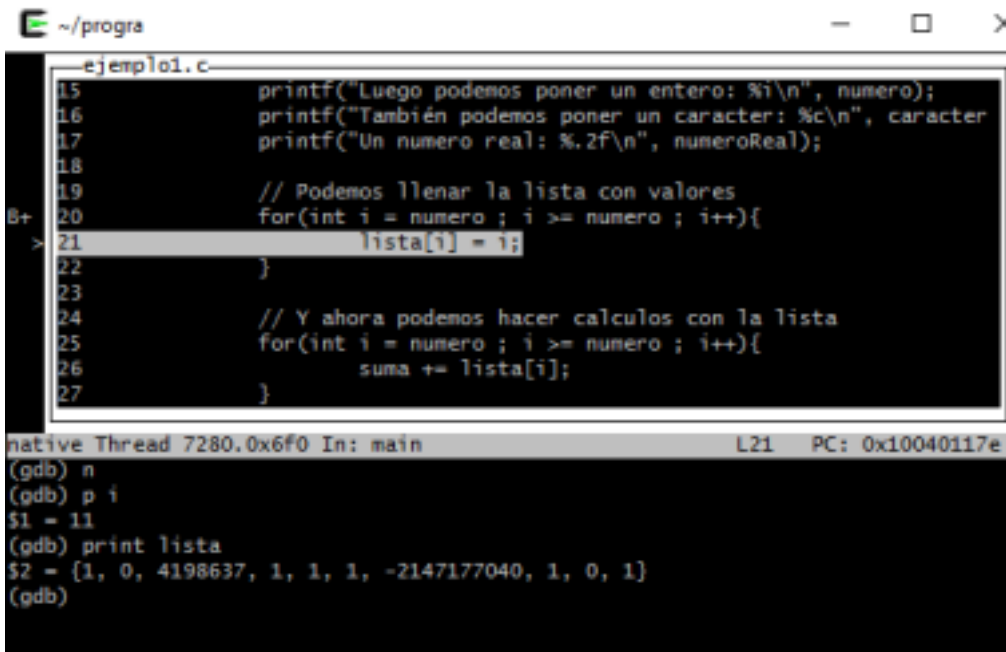
```

- Utilizamos el comando `p [variable]`, le damos enter, y nos mostrara el valor de la variable que hemos puesto, en este caso `i`.



```
~/progra
ejemplo1.c
15 printf("Luego podemos poner un entero: %i\n", numero);
16 printf("Tambi3n podemos poner un caracter: %c\n", caracter
17 printf("Un numero real: %.2f\n", numeroReal);
18
19 // Podemos llenar la lista con valores
20 for(int i = numero ; i >= numero ; i++){
21     lista[i] = i;
22 }
23
24 // Y ahora podemos hacer calculos con la lista
25 for(int i = numero ; i >= numero ; i++){
26     suma += lista[i];
27 }
native Thread 7280.0x6f0 In: main L21 PC: 0x10040117e
(gdb) n
(gdb) p i
$1 = 11
(gdb) |
```

- Utilizamos nuevamente este comando en la misma l3nea solo que ahora con la variable de lista.



```
~/progra
ejemplo1.c
15 printf("Luego podemos poner un entero: %i\n", numero);
16 printf("Tambi3n podemos poner un caracter: %c\n", caracter
17 printf("Un numero real: %.2f\n", numeroReal);
18
19 // Podemos llenar la lista con valores
20 for(int i = numero ; i >= numero ; i++){
21     lista[i] = i;
22 }
23
24 // Y ahora podemos hacer calculos con la lista
25 for(int i = numero ; i >= numero ; i++){
26     suma += lista[i];
27 }
native Thread 7280.0x6f0 In: main L21 PC: 0x10040117e
(gdb) n
(gdb) p i
$1 = 11
(gdb) print lista
$2 = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
(gdb)
```

- Si colocamos `display [variable]` muestra la variable y su valor.

```
~/progra
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }

native Thread 7280.0x6f0 In: main L21 PC: 0x10040117e
(gdb) n
(gdb) p i
$1 = 11
(gdb) print lista
$2 = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
(gdb) display i
1: i = 11
(gdb)
```

```
~/progra
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }

native Thread 7280.0x6f0 In: main L21 PC: 0x10040117e
$1 = 11
(gdb) print lista
$2 = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
(gdb) display i
1: i = 11
(gdb) display lista
2: lista = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
(gdb)
```

🌈 Si solo ponemos display nos muestra las variables sobre la línea en la que estamos y también su valor.

```
~/progra
ejemplo1.c
15     printf("Luego podemos poner un entero: %i\n", numero);
16     printf("También podemos poner un caracter: %c\n", caracter);
17     printf("Un numero real: %.2f\n", numeroReal);
18
19     // Podemos llenar la lista con valores
20     for(int i = numero ; i >= numero ; i++){
21         lista[i] = i;
22     }
23
24     // Y ahora podemos hacer calculos con la lista
25     for(int i = numero ; i >= numero ; i++){
26         suma += lista[i];
27     }

native Thread 7280.0x6f0 In: main L21 PC: 0x10040117e
(gdb) display
1: i = 11
2: lista = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
1: i = 11
2: lista = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
1: i = 11
2: lista = {1, 0, 4198637, 1, 1, 1, -2147177040, 1, 0, 1}
(gdb) |
```

### Actividad 1.

Utilizar GDB para encontrar la utilidad del programa y describir su funcionalidad.

- ✚ Hacemos la compilación con -g para que el compilador incluya la información acerca de los símbolos de nuestro programa a ejecutar. Hacemos la compilación y vemos que el programa no marca ningún error.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc -g actividad1.c -o actividad1

DELL@DESKTOP-NJVSQTU ~/progra
$
```

- ✚ Hacemos la compilación con gdb y posteriormente lo ejecutamos.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gdb ./actividad1
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ./actividad1...done.
(gdb)
```

- ✚ Corremos el programa y nos pide un número y nos da otro número como resultado.

```

Reading symbols from ./actividad1...done.
(gdb) run
Starting program: /home/DELL/progra/actividad1
[New Thread 1648.0x1210]
[New Thread 1648.0x1444]
[New Thread 1648.0x1624]
[New Thread 1648.0x1d30]
Ingresa un número: 8

El resultado es: 16
[Thread 1648.0x1624 exited with code 21]
[Thread 1648.0x1444 exited with code 21]
[Inferior 1 (process 1648) exited with code 025]
(gdb)

```

- ✚ Escribimos list para que muestre lo que está escrito en el programa. Podemos ver parte de lo que el programa hace, pero no completa, lo que se alcanza a ver es que es un programa de iteración.

```

[Inferior 1 (process 1648) exited with code 025]
(gdb) list
1      #include <stdio.h>
2
3      void main()
4      {
5          int N, CONT, AS;
6          AS=0;
7          CONT=1;
8          printf("Ingresa un número: ");
9          scanf("%i",&N);
10         while(CONT<=N)
(gdb)

```

- ✚ Escribimos q para salir y posteriormente entrar bajo gdb.

```

(gdb) list
1      #include <stdio.h>
2
3      void main()
4      {
5          int N, CONT, AS;
6          AS=0;
7          CONT=1;
8          printf("Ingresa un número: ");
9          scanf("%i",&N);
10         while(CONT<=N)
(gdb) q

DELL@DESKTOP-NJV5QTU ~/progra
$

```


- ✚ Volvemos a llamar al programa para que se ejecuta bajo gdb.

```

DELL@DESKTOP-NJV5QTU ~/progra
$ gdb actividad1
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 actividad1...done.
(gdb) |

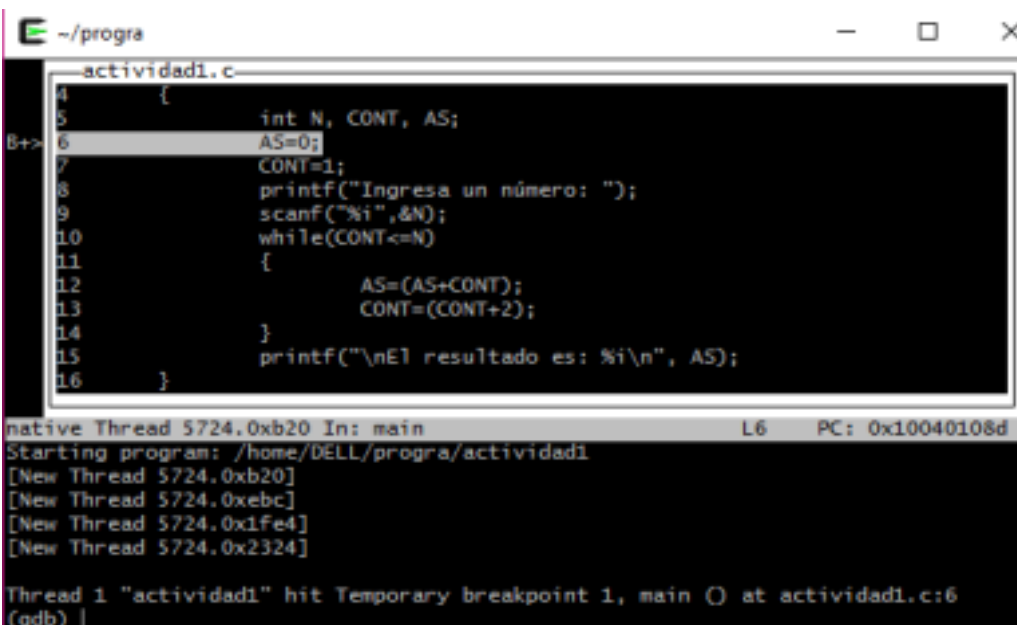
```

- Entramos a la ventanilla con él con comando CTRL+x+a.



The screenshot shows a GDB window titled '~/progra'. The main area is black with the text '[ No Source Available ]' in white. The status bar at the bottom shows 'exec No process in: L77 PC: 77' and '(gdb) |'.

- Escribimos start para que se muestre el programa completo.



The screenshot shows a GDB window titled '~/progra'. The main area displays the source code of 'actividad1.c' with line numbers 4 through 16. Line 6, 'AS=0;', is highlighted. The status bar at the bottom shows 'native Thread 5724.0xb20 In: main L6 PC: 0x10040108d', 'Starting program: /home/DELL/progra/actividad1', and 'Thread 1 "actividad1" hit Temporary breakpoint 1, main () at actividad1.c:6'. The prompt '(gdb) |' is visible.

```
4      {
5          int N, CONT, AS;
6          AS=0;
7          CONT=1;
8          printf("Ingresa un número: ");
9          scanf("%i",&N);
10         while(CONT<=N)
11         {
12             AS=(AS+CONT);
13             CONT=(CONT+2);
14         }
15         printf("\nEl resultado es: %i\n", AS);
16     }
```

- Le escribimos n para recorrer el programa línea por línea, y cuando nos encontramos en la línea de scanf nos pide ingresar un número.



```
~/progra
actividad1.c
4 {
5     int N, CONT, AS;
6     AS=0;
7     CONT=1;
8     printf("Ingresa un número: ");
9     scanf("%i",&N);
10    while(CONT<=N)
11    {
12        AS=(AS+CONT);
13        CONT=(CONT+2);
14    }
15    printf("\nEl resultado es: %i\n", AS);
16 }
```

```
native Thread 344.0x14d0 In: main L9 PC: 0x1004010a7
[New Thread 344.0x14d0]
[New Thread 344.0x149c]
[New Thread 344.0x68c]
[New Thread 344.0x20a8]

Thread 1 "actividad1" hit Temporary breakpoint 1, main () at actividad1.c:6
(gdb) n
Ingresa un número: |
```

- Después de haber ingresado el número, le damos n para continuar recorriendo el programa, y lo que sucede posteriormente es que este solo se mueve a través de las líneas que conforman al ciclo de while, esto lo hace unas cuantas veces mientras se cumple la condición que está establecida.

```
~/progra
4 {
5     int N, CONT, AS;
6     AS=0;
7     CONT=1;
8     printf("Ingresa un número: ");
9     scanf("%i",&N);
10    scanf("%i",&N);
11    while(CONT<=N)
12    {
13        AS=(AS+CONT);
14        CONT=(CONT+2);
15    }
16    printf("\nEl resultado es: %i\n", AS);
17 }
```

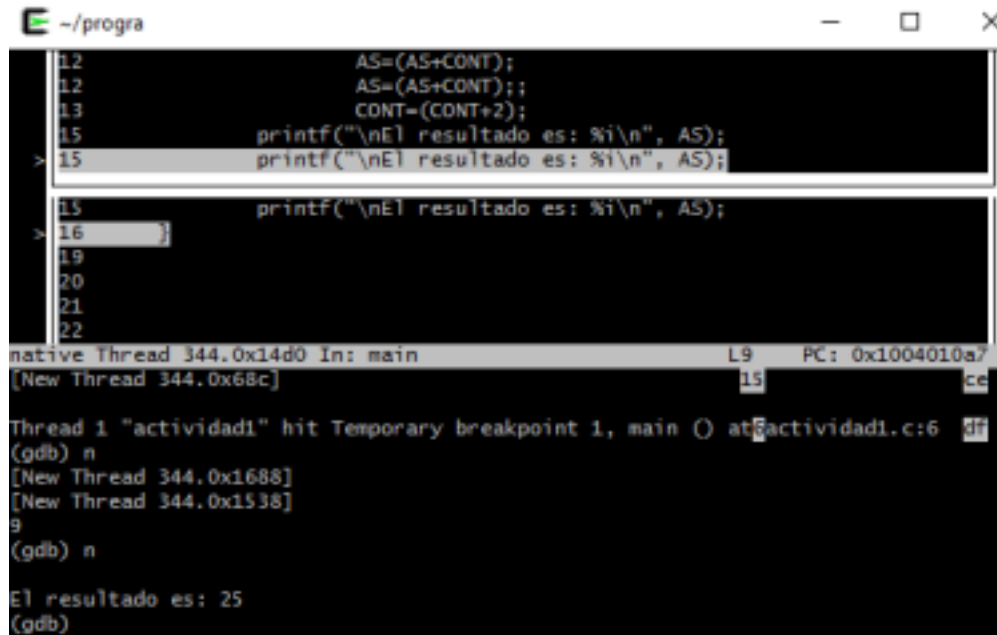
```
native Thread 344.0x14d0 In: main L9 PC: 0x1004010a7
[New Thread 344.0x68c]
[New Thread 344.0x20a8]

Thread 1 "actividad1" hit Temporary breakpoint 1, main () at actividad1.c:6
(gdb) n
[New Thread 344.0x1688]
[New Thread 344.0x1538]
9
(gdb) |
```

- Podemos ver que es un programa de iteración, donde la condición es mientras la variable llamada CONT que es una variable que acumula y que inicialmente es igual a 1, sea igual o menor a el número dado, esta incrementara en dos unidades cada vez que el ciclo se lleve a cabo. La otra variable llamada AS inicialmente es igual a cero, es un acumulador, que será igual a  $AS+CONT$ , mientras se lleve a cabo el ciclo, lo que hará que el valor de AS incremente más rápido de lo que incrementa CONT. Cuando el valor de CONT sea

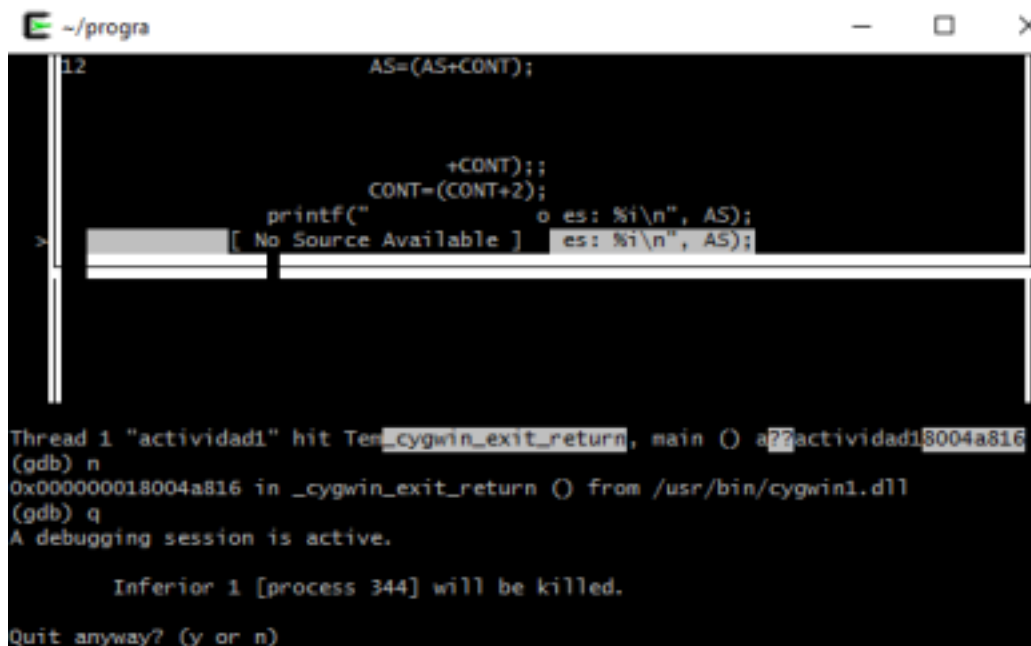
igual a el número dado entonces es cuando termina el ciclo, y el resultado que imprime es entonces el valor acumulado en AS.

- Finalmente, cuando termina de recorrer esas n veces arroja un numero como resultado.



```
~/progra
12      AS=(AS+CONT);
12      AS=(AS+CONT);
13      CONT=(CONT+2);
15      printf("\nEl resultado es: %i\n", AS);
> 15      printf("\nEl resultado es: %i\n", AS);
15      printf("\nEl resultado es: %i\n", AS);
> 16  ]
19
20
21
22
native Thread 344.0x14d0 In: main L9 PC: 0x1004010a7
[New Thread 344.0x68c] 15 ce
Thread 1 "actividad1" hit Temporary breakpoint 1, main () at actividad1.c:6
(gdb) n
[New Thread 344.0x1688]
[New Thread 344.0x1538]
9
(gdb) n
El resultado es: 25
(gdb)
```

- Por último, le damos q para salir.



```
~/progra
12      AS=(AS+CONT);
12      AS=(AS+CONT);
13      CONT=(CONT+2);
15      printf("\nEl resultado es: %i\n", AS);
> 15      printf("\nEl resultado es: %i\n", AS);
15      printf("\nEl resultado es: %i\n", AS);
> 16  ]
19
20
21
22
Thread 1 "actividad1" hit Temporary breakpoint 1, main () at actividad1.c:6
(gdb) n
0x0000000018004a816 in _cygwin_exit_return () from /usr/bin/cygwin1.dll
(gdb) q
A debugging session is active.

    Inferior 1 [process 344] will be killed.

Quit anyway? (y or n)
```

## Actividad 2.

Utilizar GDB para corregir el programa. NOTA: para compilar el código de la actividad, ejecutar:

```
$ gcc -w actividad2.c -o actividad2 -lm
```

- Primero hacemos la compilación del programa actividad2.c y para esto utilizaremos lo siguiente: `$ gcc -w actividad2.c -o actividad2 -lm`.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc -w actividad2.c -o actividad2 -lm

DELL@DESKTOP-NJVSQTU ~/progra
$
```

- Vemos que hace la compilación sin ningún problema. Entonces vamos a ejecutar el programa.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ ./actividad2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=
```

- Al ejecutarlo nos pide un número, pero después de ingresar el numero el programa no hace prácticamente nada.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ ./actividad2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=4

DELL@DESKTOP-NJVSQTU ~/progra
$
```

- Volvemos a compilar el programa, pero esta vez con -g para que guarde la información de los símbolos utilizados en gdb y lo ejecutamos con ayuda de gdb.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc -g actividad2.c -o actividad2

DELL@DESKTOP-NJVSQTU ~/progra
$ gdb ./actividad2
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ./actividad2...done.
(gdb)
```

- Le damos run para correr el programa.

```
Reading symbols from ./actividad2...done.
(gdb) run
Starting program: /home/DELL/progra/actividad2
[New Thread 4824.0x4d4]
[New Thread 4824.0x8bc]
[New Thread 4824.0x700]
[New Thread 4824.0x1a88]
Ingrese cuántos términos calcular de la serie: X^K/K!
```

- Al correrlo nos pide nuevamente un número y nuevamente no hace nada.

```
Ingrese cuántos términos calcular de la serie: X^K/K!
N=5
[Thread 4824.0x8bc exited with code 3221225477]
[Thread 4824.0x700 exited with code 3221225477]
[Inferior 1 (process 4824) exited with code 030000000005]
```

- Para ver el programa le damos list. Y aparece lo siguiente:

```
(gdb) list
1  #include <stdio.h>
2  #include <math.h>
3
4  void main()
5  {
6      int K, AP, N;
7      double X, AS;
8      printf("Ingrese cuántos términos calcular de la serie: X^K/K!");
9      printf("\nN=");
10     scanf("%i",N);
(gdb)
```

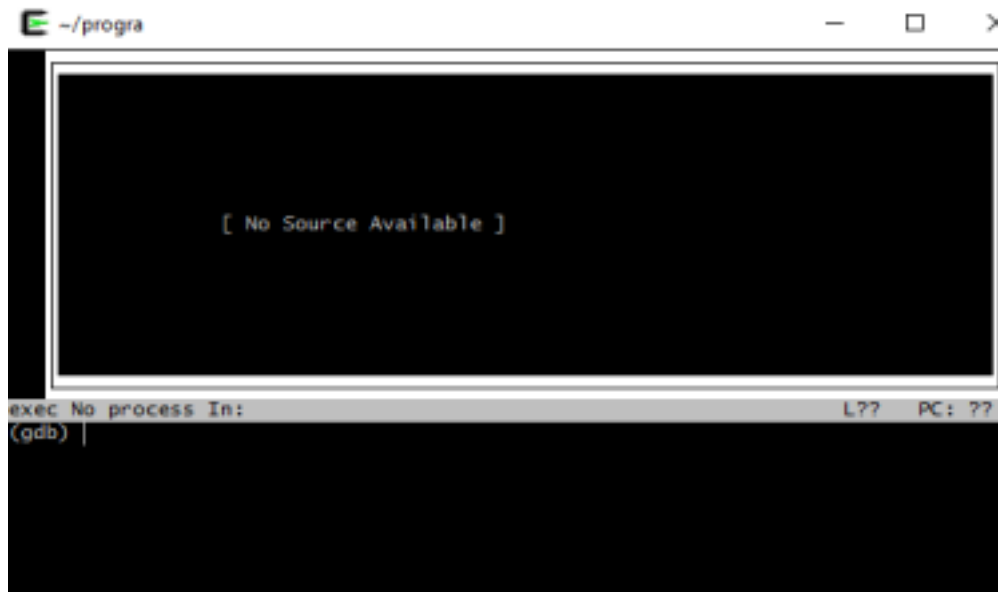
- ✚ Escribimos q para salir.

```
(gdb) q
DELL@DESKTOP-NJVSQTU ~/progra
$ |
```

- ✚ Llamamos al programa para ejecutarlo bajo gdb.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gdb actividad2
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 actividad2...done.
(gdb)
```

- ✚ Abrimos la ventanilla con CTRL+x+a.



- ✚ Le damos start, para que muestre el programa en la ventanilla.

```

~/progra
actividad2.c
5      {
6          int K, AP, N;
7          double X, AS;
8->      printf("Ingrese cuántos términos calcular de la serie: X^N");
9          printf("\nN=");
10         scanf("%i", &N);
11         printf("X=");
12         scanf("%lf", &X);
13         K=0;
14         AP=1;
15         AS=0;
16         while(K<=N)
17         {
native Thread 3440.0x13f4 In: main L8 PC: 0x10040108d
Starting program: /home/DELL/progra/actividad2
[New Thread 3440.0x13f4]
[New Thread 3440.0xc3c]
[New Thread 3440.0xfd0]
[New Thread 3440.0x1314]
Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) |

```

🔧 Recorremos el programa con ayuda de next.

```

~/progra
5      {
6          int K, AP, N;
7          double X, AS;
8          printf("Ingrese cuántos términos calcular de la serie: X^N");
9->      printf("\nN=");
9          printf("\nN=");
10->      scanf("%i", &N);
12         scanf("%lf", &X);
13         K=0;
14         AP=1;
15         AS=0;
16         while(K<=N)
17         {
native Thread 5360.0x1d24 In: main L9 PC: 0x100401099
[New Thread 5360.0x1d24]
[New Thread 5360.0x720]
[New Thread 5360.0x470]
[New Thread 5360.0x16d8]
Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) n
Ingrese cuántos términos calcular de la serie: X^K/K!
(gdb) |

```

🔧 Cuando nos encontramos en la parte donde pide ingresar el número el programa ya no corre.

```

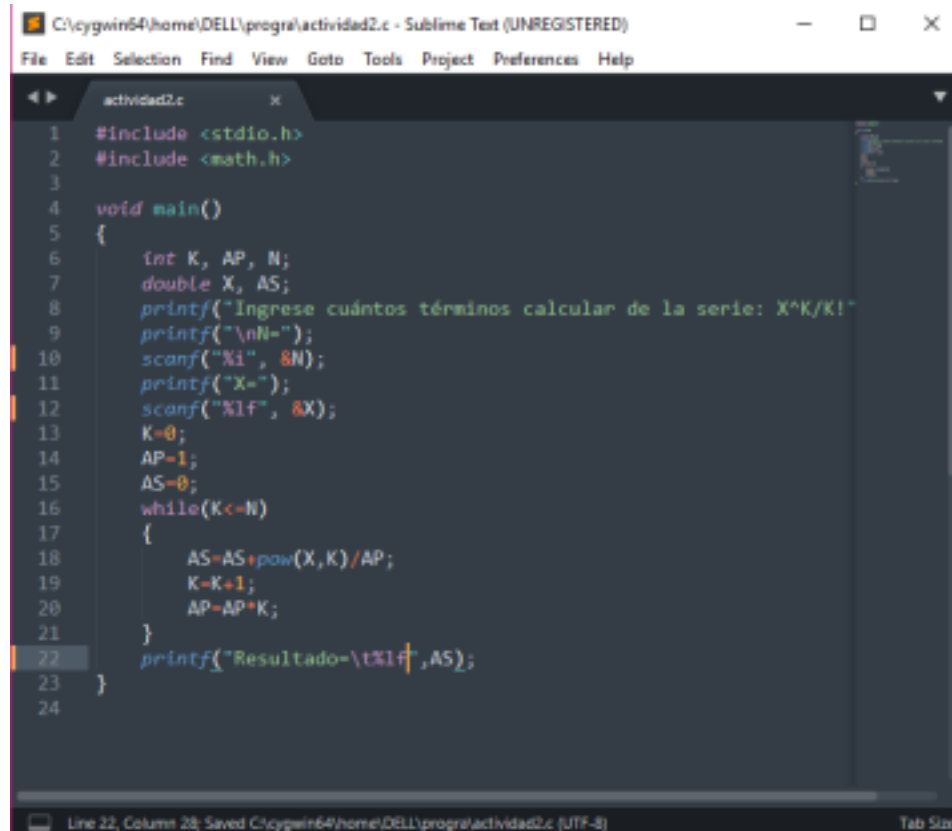
~/progra
6          int K, AP, N;
1          #include <stdio.h>
2          #include <math.h>
3
4          void main()
5          {
6              double X, AS;
7              int K, AP, N; se cuántos términos calcular de la serie: X^N
8              double X, AS;
9->      printf("Ingrese cuántos términos calcular de la serie: X^N");
10->      printf("\nN=");
10         scanf("%i", &N);
11         printf("X=");
12         scanf("%lf", &X);
13         K=0;
[New Thread 3968.0x171c]
exec No process in: L77 PC: ??
N=3
[Thread 3968.0x1bc4 exited with code 3221225477]
[Thread 3968.0x1de8 exited with code 3221225477]
[Inferior 1 (process 3968) exited with code 0300000000005]
(gdb) n
The program is not being run.
The program is not being run.
(gdb) |

```

- ✚ Por último, presionamos q para salir.

- ✚ Podemos ver que, aunque el programa corre y pide los datos estos no se guardan ni hacen las operaciones correspondientes y esto ocurre debido a los errores de sintaxis que se encuentran en el programa. Entonces corregiré estos errores en el editor de texto.
- ✚ Inicialmente el programa se encuentra estructurado así:

- ✚ Al hacer la corrección queda de esta forma.



```
1 #include <stdio.h>
2 #include <math.h>
3
4 void main()
5 {
6     int K, AP, N;
7     double X, AS;
8     printf("Ingrese cuántos términos calcular de la serie: X^K/K!\n");
9     printf("\nN=");
10    scanf("%i", &N);
11    printf("X=");
12    scanf("%lf", &X);
13    K=0;
14    AP=1;
15    AS=0;
16    while(K<=N)
17    {
18        AS=AS+pow(X,K)/AP;
19        K=K+1;
20        AP=AP*K;
21    }
22    printf("Resultado=\tXlf", AS);
23 }
24
```

- La función pow utilizado en este programa significa:  
Pow (base, n); (pow: eleva la base a n-ésima potencia; n>=0)
- Volvemos a hacer la compilación del programa con:  
\$ gcc -w actividad2.c -o actividad2 -lm

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gcc -w actividad2.c -o actividad2 -lm
DELL@DESKTOP-N3V5QTU ~/progra
```

- Y vemos que guarda el numero ingresado y que continua con el proceso del programa, proceso que no se pudo ver anteriormente debido a los errores de las sintaxis.

```
DELL@DESKTOP-N3V5QTU ~/progra
$ ./actividad2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=3
X=5
Resultado=      39.333333
DELL@DESKTOP-N3V5QTU ~/progra
$
```

- Compilamos el programa con -g para que guarde la información de los símbolos del programa en gdb.

```
DELL@DESKTOP-N3V5QTU ~/progra
$ gcc -g actividad2.c -o actividad2
DELL@DESKTOP-N3V5QTU ~/progra
$
```

- Mandamos a llamar al programa para que se ejecute bajo gdb.

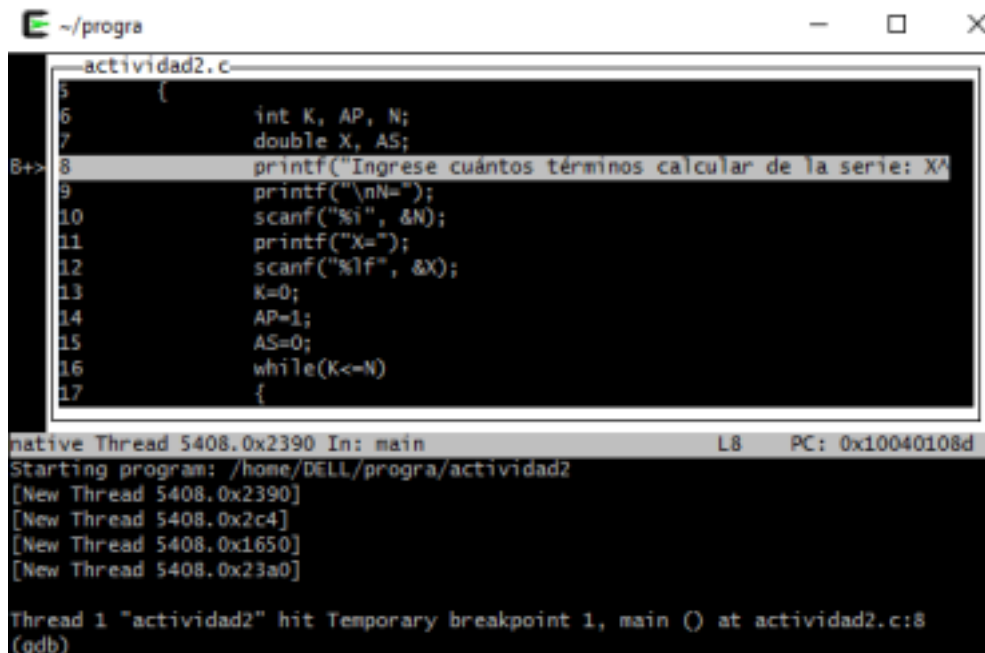


```

DELL@DESKTOP-NJVSQTU ~/progra
$ gdb actividad2
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 actividad2...done.
(gdb)

```

- ✚ Abrimos la ventanilla con CTRL+x+a y le escribimos start para ver el programa en la ventanilla.

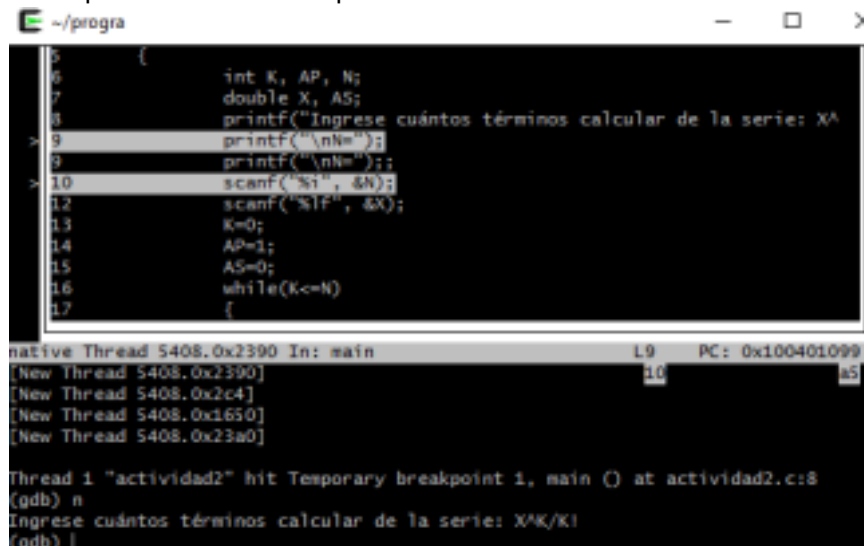


```

~/progra
actividad2.c
5 {
6     int K, AP, N;
7     double X, AS;
8     printf("Ingrese cuántos términos calcular de la serie: X^
9     printf("\nN=");
10    scanf("%i", &N);
11    printf("X=");
12    scanf("%lf", &X);
13    K=0;
14    AP=1;
15    AS=0;
16    while(K<=N)
17    {
native Thread 5408.0x2390 In: main L8 PC: 0x10040108d
Starting program: /home/DELL/progra/actividad2
[New Thread 5408.0x2390]
[New Thread 5408.0x2c4]
[New Thread 5408.0x1650]
[New Thread 5408.0x23a0]
Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb)

```

- ✚ Escribimos n para avanzar línea por línea.



```

~/progra
actividad2.c
5 {
6     int K, AP, N;
7     double X, AS;
8     printf("Ingrese cuántos términos calcular de la serie: X^
9     printf("\nN=");
10    scanf("%i", &N);
11    printf("X=");
12    scanf("%lf", &X);
13    K=0;
14    AP=1;
15    AS=0;
16    while(K<=N)
17    {
native Thread 5408.0x2390 In: main L9 PC: 0x100401099
[New Thread 5408.0x2390]
[New Thread 5408.0x2c4]
[New Thread 5408.0x1650]
[New Thread 5408.0x23a0]
Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) n
Ingrese cuántos términos calcular de la serie: X^K/K!
(gdb)

```

- ✚ Cuando nos situamos en la línea donde se pide ingresar un número, ahora si guarda el número y sigue con la siguiente instrucción que es ingresar otro número.

```

~/progra
7      double X, AS;
8      printf("Ingrese cuántos términos calcular de la serie: X^N\n");
9      printf("\nN=");
10     scanf("%i", &N);
11     printf("X=");
12     scanf("%lf", &X);
13     K=0; while(K<=N)
14     {
15
native Thread 2064.0x1524 In: main          L9      PC: 0x100401099
[New Thread 2064.0x1524]
[New Thread 2064.0x290]
[New Thread 2064.0xac4]
[New Thread 2064.0xb2c]

Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) n
Ingrese cuántos términos calcular de la serie: X^K/K!
N=3
X=4
(gdb) |

```

Después vemos que se ejecuta el ciclo de while hasta que ya no cumpla la condición.

```

~/progra
7      double X, AS;
8      printf("Ingrese cuántos términos calcular de la serie: X^N\n");
9      printf("\nN=");
10     scanf("%i", &N);
11     printf("X=");
12     scanf("%lf", &X);
13     K=0; while(K<=N)
14     {
15         AP=1;
16         AS=0; AS=AS+pow(X,K)/AP;
17         while(K<=N)+1;
18         AP=AP*K;
19         AS=AS+pow(X,K)/AP;
20         K=K+1; adom=\t%lf", AS);
21         AP=AP*K;
22     }

native Thread 5012.0x1cc0 In: main          L9      PC: 0x100401099
Temporary breakpoint 1 at 0x10040108d: file actividad2.c, line 8.
Starting program: /home/DELL/progra/actividad2
[New Thread 5012.0x1cc0]
[New Thread 5012.0x1fda]
[New Thread 5012.0xa6c]
[New Thread 5012.0x2ec]

Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) n
Ingrese cuántos términos calcular de la serie: X^K/K!
N=3
X=4
(gdb) |

```

```

~/progra
8      printf("Ingrese cuantos terminos calcular de la serie: X^K/K!");
      AS=AS+pow(X,K)/AP;
      K=K+1;
      AP*=K;
      5);
Thread 5056.0 1608 in: main
L9 PC: 0x100401099
1dad2.c, 110e 8.
ram: [ No Source Available ] vidad2
[New Thread 5056.0x1608]
[New Thread 5056.0x1d4c]
[New Thread 5056.0xe2c]
Thread 1 "actividad2" hit Temporary breakpoint 1, main () at actividad2.c:8
(gdb) m
Ingrese cuantos terminos calcular de la serie: X^K/K!
3
4
0x000000018004a816 in _cygwin_exit_return () from /usr/bin/cygwin1.dll
(gdb)

```

- Lo que podemos ver cuando el programa ya está corregido es que está diseñado para dar el resultado de una sumatoria de una serie donde el primer número será la potencia a la que se elevará el segundo número ingresado y este será dividido entre la factorial del primer número ingresado y esto se hará sucesivamente disminuyendo de uno en uno el primer número hasta llegar a cero. Cuando el primer número llega a cero el resultado acumulado de las operaciones anteriores es guardado en una variable acumulador en este caso AS, al final es lo que se imprime.

### Actividad 3.

Utilizar GDB para corregir el programa.

- Primero hacemos la compilación del programa. Y vemos que no tiene problema alguna su compilación.

```

DELL@DESKTOP-N3V5QTU ~/progra
$ gcc actividad3.c -o actividad3

DELL@DESKTOP-N3V5QTU ~/progra
$ |

```

- Después corremos el programa, vemos que nos pide ingresar un número y el resultado que nos da no tiene nada que ver con el numero ingresado, además de que es erróneo.

```

DELL@DESKTOP-N3V5QTU ~/progra
$ ./actividad3
Ingrese un número:
4
El factorial de -1 es 0.

DELL@DESKTOP-N3V5QTU ~/progra
$

```

- Hacemos nuevamente la compilación del programa utilizando -g para se guarde la información de los símbolos utilizados en gdb.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc -g actividad3.c -o actividad3

DELL@DESKTOP-NJVSQTU ~/progra
$
```

- ✚ Corremos el programa con gdb.

```
DELL@DESKTOP-NJVSQTU ~/progra
$ gdb ./actividad3
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 ./actividad3...done.
(gdb)
```

- ✚ Le ponemos run. Y vemos lo que hace.

```
Reading symbols from ./actividad3...done.
(gdb) run
Starting program: /home/DELL/progra/actividad3
[New Thread 8816.0x878]
[New Thread 8816.0xd8]
[New Thread 8816.0x19f4]
[New Thread 8816.0x1c1c]
Ingrese un número:
4
El factorial de -1 es 0.
[Thread 8816.0x1c1c exited with code 0]
[Thread 8816.0x19f4 exited with code 0]
[Inferior 1 (process 8816) exited normally]
(gdb)
```

- ✚ Le ponemos list para ver la estructura del programa.

```
(gdb) list
1      #include <stdio.h>
2
3      int main()
4      {
5          int numero;
6
7          printf("Ingrese un número:\n");
8          scanf("%i",&numero);
9
10         long int resultado = 1;
(gdb) |
```

- ✚ Nos salimos escribiendo q.

```
10         long int resultado = 1;
(gdb) q
```

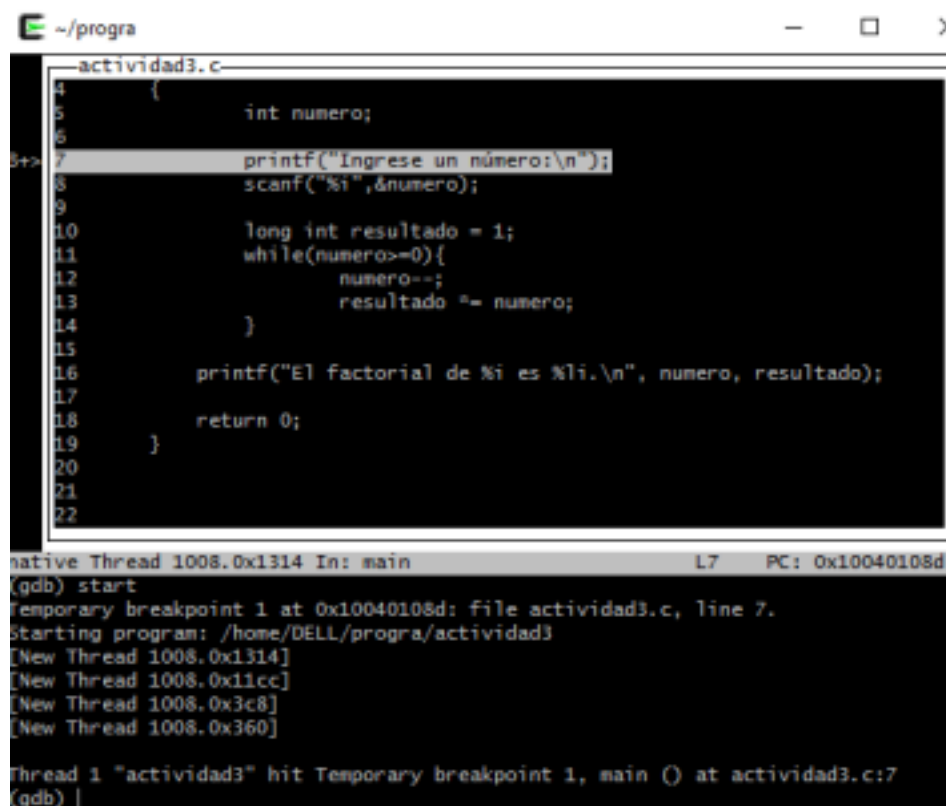
- ✚ Ahora llamaremos al programa para que se ejecute bajo gdb.

```

DELL@DESKTOP-NJVSQTU ~/progra
$ gdb actividad3
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 actividad3...done.
(gdb)

```

- ✚ Escribimos CTRL+x+a para entrar a la ventanilla. Después escribimos start para ver al programa dentro de la ventanilla.



The screenshot shows a GDB window titled '~/.progra'. The main pane displays the source code of 'actividad3.c' with line numbers 4 through 22. Line 7, which contains the printf statement 'printf("Ingrese un número:\n");', is highlighted. The bottom pane shows the GDB console output, including the command '(gdb) start', the location of a temporary breakpoint at line 7, and the start of the program execution. The console also lists several new threads created during execution.

```

~/.progra
actividad3.c
4      {
5          int numero;
6
7      printf("Ingrese un número:\n");
8      scanf("%i",&numero);
9
10     long int resultado = 1;
11     while(numero>=0){
12         numero--;
13         resultado *= numero;
14     }
15
16     printf("El factorial de %i es %li.\n", numero, resultado);
17
18     return 0;
19 }
20
21
22
native Thread 1008.0x1314 In: main L7 PC: 0x10040108d
(gdb) start
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.
Starting program: /home/DELL/progra/actividad3
[New Thread 1008.0x1314]
[New Thread 1008.0x11cc]
[New Thread 1008.0x3c8]
[New Thread 1008.0x360]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) |

```

- ✚ Le damos next o n para recorrer línea por línea el programa. Hasta llegar a la línea donde nos pide que ingresemos un número.

```
~/progra
4 {
5     int numero;
6
7-> 7     printf("Ingrese un número:\n");
7     printf("Ingrese un número:\n");
> 8     scanf("%i",&numero);
10     long int resultado = 1;
11     while(numero>=0){
12         numero--;
13         resultado *= numero;
14     }
15
16     printf("El factorial de %i es %li.\n", numero, resultado);
17
18     return 0;
19 }
20
21
22

native Thread 1276.0xb34 In: main L7 PC: 0x10040108d
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7. 99
Starting program: /home/DELL/progra/actividad3
[New Thread 1276.0xb34]
[New Thread 1276.0xf78]
[New Thread 1276.0x1228]
[New Thread 1276.0x11d0]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) n
Ingrese un número:
4
```

El programa ejecuta la condición de while.

```
~/progra
5     int numero;
6
7-> 7     printf("Ingrese un número:\n");
7     printf("Ingrese un número:\n");
> 8     scanf("%i",&numero);
8     scanf("%i",&numero); 1;
11     while(numero>=0){
10     long int resultado = 1;
> 11     while(numero>=0){ *= numero;
14     }
15
16     printf("El factorial de %i es %li.\n", numero, resultado);
17
18     return 0;
19 }
20
21
22

native Thread 1276.0xb34 In: main L7 PC: 0x10040108d
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7. 99
[New Thread 1276.0xf78] 11 b4
[New Thread 1276.0x1228]
[New Thread 1276.0x11d0]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) n
Ingrese un número:
[New Thread 1276.0x1ab0]
[New Thread 1276.0x1930]
4
(gdb) |
```

Resulta que al darle n para seguir viendo el proceso que realiza el programa, este solo recorre el ciclo indeterminadamente.

Salimos de la ventanilla escribiendo q.

```
~/progra
5      int numero;
6      #include <stdio.h>
7      int main()
8      {
9          printf("Ingrese un número:\n");
10         printf("Ingrese un número:\n");
11         scanf("%i",&numero);
12         scanf("%i",&numero); 1;
13         while(numero>=0){
14             long int resultado = 1;
15             while(numero>=0){ *= numero;
16                 numero--;
17                 resultado *= numero;
18             }
19             printf("El factorial de %i es %li.\n", numero, resultado);
20         }
21         return 0;
22     }

Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.
[New Thread 1276.0x1f78]
[New Thread 1276.0x1ab0]
[New Thread 1276.0x1930]
(gdb) n
(gdb) q
A debugging session is active.

Inferior 1 [process 1276] will be killed.

Quit anyway? (y or n) |
```

- Ahora corregiremos el programa en el editor de texto. Inicialmente el programa se encuentra de la siguiente manera:

```
C:\cygwin64\home\DELL\progra\actividad3.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

actividad2.c x actividad3.c
1  #include <stdio.h>
2
3  int main()
4  {
5      int numero;
6
7      printf("Ingrese un número:\n");
8      scanf("%i",&numero);
9
10     long int resultado = 1;
11     while(numero>=0){
12         numero--;
13         resultado *= numero;
14     }
15
16     printf("El factorial de %i es %li.\n", numero, resultado);
17
18     return 0;
19 }
20
21

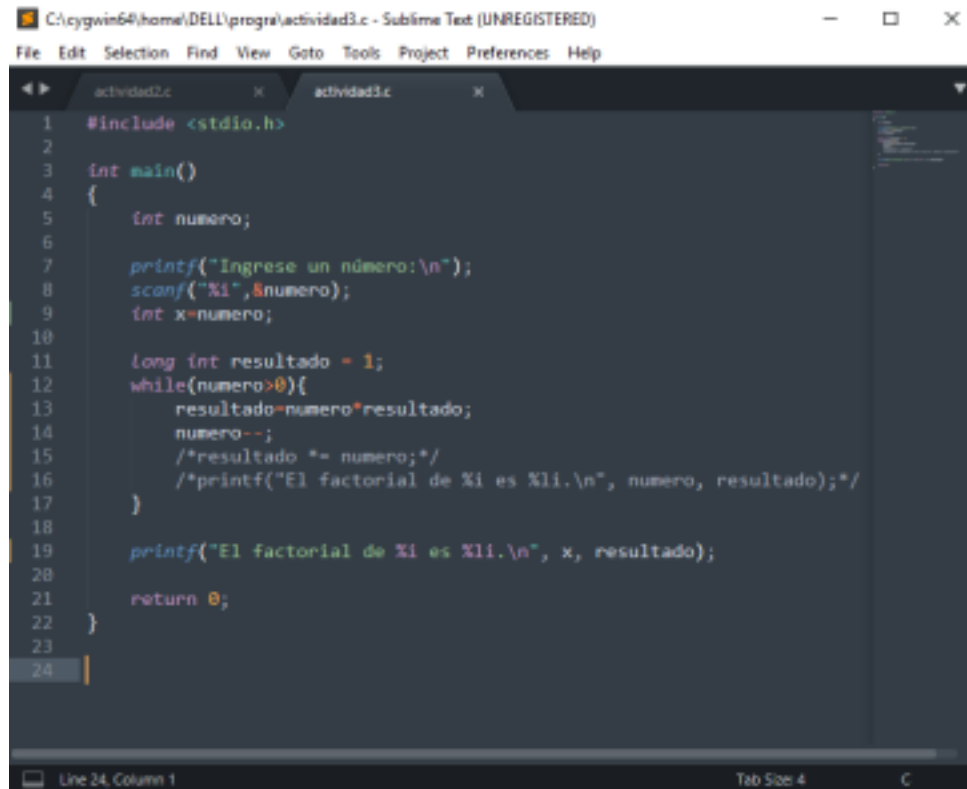
Line 8, Column 25    Tab Size: 4    C
```

- Vemos que tiene errores de sintaxis.
- Long int indica que la variable es un entero de longitud de 32 bits.

```
10     long int resultado = 1;
```

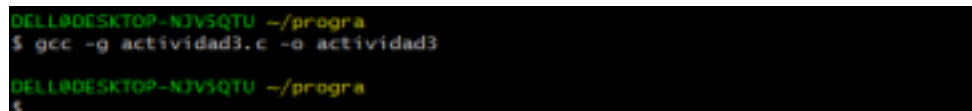
- Corrigiendo los errores de sintaxis el programa queda de la siguiente manera:





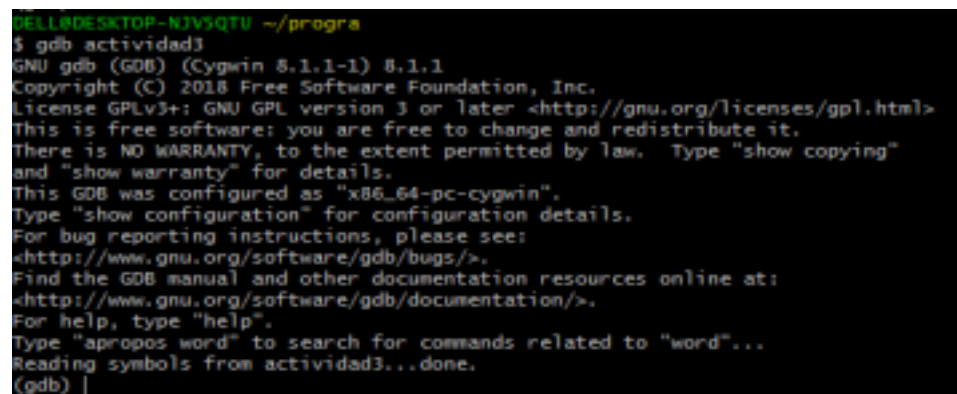
```
1 #include <stdio.h>
2
3 int main()
4 {
5     int numero;
6
7     printf("Ingrese un número:\n");
8     scanf("%i",&numero);
9     int x=numero;
10
11     long int resultado = 1;
12     while(numero>0){
13         resultado=numero*resultado;
14         numero--;
15         /*resultado *= numero;*/
16         /*printf("El factorial de %i es %li.\n", numero, resultado);*/
17     }
18
19     printf("El factorial de %i es %li.\n", x, resultado);
20
21     return 0;
22 }
23
24
```

- ✚ Volvemos a compilar el programa con -g para que guarde la información de los símbolos en gdb.



```
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc -g actividad3.c -o actividad3
DELL@DESKTOP-NJVSQTU ~/progra
$
```

- ✚ Llamamos al programa para que se ejecute bajo gdb.



```
DELL@DESKTOP-NJVSQTU ~/progra
$ gdb actividad3
GNU gdb (GDB) (Cygwin 8.1.1-1) 8.1.1
Copyright (C) 2018 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-pc-cygwin".
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 actividad3...done.
(gdb) |
```

- ✚ Abrimos nuevamente la ventanilla con CTRL+x+a y le escribimos start para ver el programa en la ventanilla.

```
~/progra
actividad3.c
4      {
5          int numero;
6
7      printf("Ingrese un número:\n");
8      scanf("%i",&numero);
9      int x=numero;
10
11      long int resultado = 1;
12      while(numero>0){
13          resultado=numero*resultado;
14          numero--;
15          /*resultado *= numero;*/
16          /*printf("El factorial de %i es %li.\n", numero, resultado);*/
17      }
18
19      printf("El factorial de %i es %li.\n", x, resultado);
20
21      return 0;
22  }
```

```
native Thread 888.0x131c In: main L7 PC: 0x10040108d
(gdb) start
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.
Starting program: /home/DELL/progra/actividad3
[New Thread 888.0x131c]
[New Thread 888.0x2314]
[New Thread 888.0x1f00]
[New Thread 888.0x20b8]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb)
```

🚧 Recorremos el programa línea por línea con ayuda de next.

```
~/progra
4      {
5          int numero;
6
7      printf("Ingrese un número:\n");
7      printf("Ingrese un número:\n");
> 8      scanf("%i",&numero);
10
11      long int resultado = 1;
12      while(numero>0){
13          resultado=numero*resultado;
14          numero--;
15          /*resultado *= numero;*/
16          /*printf("El factorial de %i es %li.\n", numero, resultado);*/
17      }
18
19      printf("El factorial de %i es %li.\n", x, resultado);
20
21      return 0;
22  }
```

```
native Thread 7996.0x21c8 In: main L7 PC: 0x10040108d
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.
Starting program: /home/DELL/progra/actividad3
[New Thread 7996.0x21c8]
[New Thread 7996.0x15b4]
[New Thread 7996.0x221c]
[New Thread 7996.0x1d4c]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) n
Ingrese un número:
(gdb)
```

🚧 Llegamos a la línea donde nos pide ingresar un número y lo ingresamos.

```
~/progra
5         int numero;
6
7         printf("Ingrese un número:\n");
8         scanf("%i",&numero);
9         int x=numero;ltado = 1;
12        while(numero>0){
11            long int resultado = 1;o*resultado;
12            while(numero>0){;
13                resultado=numero*resultado;
14            numero--;"El factorial de %i es %li.\n", numero, re
17        }
18
19        printf("El factorial de %i es %li.\n", x, resultado);
20
21        return 0;
22    }
```

Native Thread 5756.0x1534 In: main L7 PC: 0x10040108d  
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.  
Starting program: /home/DELL/progra/actividad3  
[New Thread 5756.0x1534]  
[New Thread 5756.0x5a8]  
[New Thread 5756.0xd68]  
[New Thread 5756.0x11d0]  
Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7  
(gdb) n  
Ingrese un número:  
5  
(gdb)

- ✚ Vemos que realiza el ciclo de while mientras se cumpla la condición.
- ✚ Finalmente, el programa imprime el resultado.

```
~/progra
14            numero--;"El factorial de %i es %li.\n", numero, re
17        }
18
19        printf("El factorial de %i es %li.\n", x, resultado);
20
21        printf("El factorial de %i es %li.\n", x, resultado);
22        return 0; In: main L7 PC: 0x10040108d
Temporary breakpoint 1 at 0x10040108d: file actividad3.c, line 7.
23
24
25
26
27
28
29
30
31
Starting program: /home/DELL/progra/actividad3
[New Thread 5756.0x5a8]
[New Thread 5756.0xd68]
[New Thread 5756.0x11d0]
Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) n
Ingrese un número:
5
[New Thread 5756.0x2388]
El factorial de 5 es 120.
(gdb)
```

- ✚ Salimos del programa escribiendo q.

```
1 6
2
3     int main()
4     {
5         int numero;
6
7         printf("Ingrese un número:\n");
8         scanf("%i",&numero);
9         int x=numero;
10
11        long int resultado = 1;
12        while(numero>0){
13            resultado=numero*resultado;
14            numero--;
15            /*resultado *= numero;*/
16            /*printf("El factorial de %i es %li.\n", numero, re
17        }
18
19        printf("El factorial de %i es %li.\n", x, resultado);
20
Starting program: /home/DELL/progra/actividad3
[New Thread 5756.0x5a8]
[New Thread 5756.0xd68]
[New Thread 5756.0x11d0]

Thread 1 "actividad3" hit Temporary breakpoint 1, main () at actividad3.c:7
(gdb) n
Ingrese un número:
5
[New Thread 5756.0x2388]
El factorial de 5 es 120.
(gdb) q
```

Este programa da como resultado la factorial del número ingresado, para poder imprimir el numero ingresado este se guardará en dos variables, una de ellas se queda con el valor del número ingresado y el otro se utiliza en el ciclo y va decreciendo de uno en uno. La variable acumuladora que guarda el resultado final es inicialmente igual a uno y después se estará multiplicando por la variable que guarda el numero ingresado que se encuentra en el ciclo de while. El orden que tenía estaba incorrecto ya que primero tenía que hacer la multiplicación de los numero y después la variable número se tenía que ir disminuyendo de uno en uno, el otro factor era que la variable número no tenía que ser igual a 0 sino solo mayor que cero.

### Conclusión:

Esta practica no resulta ser difícil, pero si algo larga, pero vale la pena aprender cómo utilizar la herramienta de GDB. Utilizar GDB resulta ser muy útil para detectar los errores que se pueden tener en un programa, dado que se puede visualizar el programa escrito, y se puede ir viendo paso por paso como se va ejecutando. De esta manera podemos ir comprendiendo lo que sucede en el programa y hacer los cambios necesarios para que el programa realice el proceso que queremos.