	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de Programación

Grupo: 3; Bloque: 135

Práctica(s): 10

Intearante(s): Bazaldúa Morales Vanessa
Torres Alcántara Alan Eliezer

No. de Equipo de cómputo Somalia 42

No. de Lista o 9032; #6-8166

Semestre: 2020 - 1

Fecha de 21/10/2019

Observaciones:

DEPURACIÓN DE PROGRAMAS

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.

Introducción:

Depurar un programa significa someterlo a un ambiente de ejecución controlado por medio de herramientas dedicadas a ello. Este ambiente permite conocer exactamente el flujo de ejecución del programa, el valor que las variables adquieren, la pila de llamadas a funciones, entre otros aspectos. Es importante poder compilar el programa sin errores antes de depurarlo.

Antes de continuar, es necesario conocer las siguientes definiciones ya que son parte latente del proceso de Desarrollo de Software:

- *Error:* Se refiere a una acción humana que produce o genera un resultado incorrecto.
- *Defecto (fault):* Es la manifestación de un error en el software. Un defecto es encontrado porque causa una falla.
- *Falla (failure):* Es una desviación del servicio o resultado esperado.

La depuración de un programa es útil cuando:

- *Se desea optimizar el programa:* no basta que el programa se pueda compilar y se someta a pruebas que demuestren que funciona correctamente. Debe realizarse un análisis exhaustivo del mismo en ejecución para averiguar cuál es su flujo de operación y encontrar formas de mejorarlo (reducir el código, utilizar menos recursos llegando a los mismos resultados, hacer menos rebuscado al algoritmo), o bien, encontrar puntos donde puede fallar con ciertos tipos de entrada de datos.
- *El programa tiene algún fallo:* el programa no muestra los resultados que se esperan para cierta entrada de datos debido a que el programador cometió algún error durante el proceso de diseño. Muchas veces encontrar este tipo de fallos suele ser difícil, ya sea porque la percepción del programador no permite encontrar la falla en su diseño o porque la errata es muy pequeña, pero crucial. En este caso es de mucha utilidad conocer paso a paso cómo se ejecutan las estructuras de control, qué valor adquieren las variables, etc.
- *El programa tiene un error de ejecución o defecto:* cuando el programa está ejecutándose, éste se detiene inesperadamente. Suele ocurrir por error en el diseño o implementación del programa en las

que no se contemplan las limitaciones del lenguaje de programación o el equipo donde el programa se ejecuta. Como el programa se detiene inesperadamente, no se conoce la parte del programa donde se provoca el defecto, teniendo que recurrir a la depuración para encontrarlo. El más común de este tipo de defecto es la “violación de segmento”.

Algunas funciones básicas que tienen en común la mayoría de los depuradores son las siguientes:

- *Ejecutar el programa:* se procede a ejecutar el programa en la herramienta de depuración ofreciendo diversas opciones para ello.
- *Mostrar el código fuente del programa:* muestra cuál fue el código fuente del programa con el número de línea con el fin de emular la ejecución del programa sobre éste, es decir, se indica qué parte del código fuente se está ejecutando a la hora de correr el programa.
- *Punto de ruptura:* también conocido por su traducción al inglés breakpoint, sirve para detener la ejecución del programa en algún punto indicado previamente por medio del número de línea. Como la ejecución del programa es más rápida de lo que podemos visualizar y entender, se suelen poner puntos de ruptura para conocer ciertos parámetros de la ejecución como el valor de las variables en determinados puntos del programa. También sirve para verificar hasta qué punto el programa se ejecuta sin problemas y en qué parte podría existir el error, esto es especialmente útil cuando existe un error de ejecución.
- *Continuar:* continúa con la ejecución del programa después del punto de ruptura.
- *Ejecutar la siguiente instrucción:* cuando la ejecución del programa se ha detenido por medio del depurador, esta función permite ejecutar una instrucción más y detener el programa de nuevo. Esto es útil cuando se desea estudiar detalladamente una pequeña sección del programa. Si en la ejecución existe una llamada a función se ingresará a ella.
- *Ejecutar la siguiente línea:* es muy similar a la función anterior, pero realizará todas las instrucciones necesarias hasta llegar a la siguiente línea de código. Si en la ejecución existe una llamada a función se ignorará.
- *Ejecutar la instrucción o línea anterior:* deshace el efecto provocado por alguna de las funciones anteriores para repetir una sección del programa.
- *Visualizar el valor de las variables:* permite conocer el valor de alguna o varias variables.

Ejemplo

```c
#include <stdio.h>

int main(int argc, char * argv[]) {

    // Asignamos variables
    int numero = 10;
    int lista[numero];
    char caracter = 'B';
    float numeroReal = 89.8;
    long int suma = 0;
    double promedio;

    // Mostramos texto y valores
    printf("Primero texto solo\n");
    printf("Luego podemos poner un entero: %i\n", numero);
    printf("También podemos poner un caracter: %c\n", caracter);
    printf("Un numero real: %.2f\n", numeroReal);

    // Podemos llenar la lista con valores
    for(int i = numero ; i >= numero ; i++){
        lista[i] = i;
    }

    // Y ahora podemos hacer calculos con la lista
    for(int i = numero ; i >= numero ; i++){
        suma += lista[i];
    }
    promedio = suma / numero;
    printf("La suma es: %li\n", suma);
    printf("El promedio es: %lf\n",promedio);

    return 0;
}
```

gcc -g

```
Tonga51:Desktop fp03alu48$ gcc ejemplo1.c -o ejemplo1_
Tonga51:Desktop fp03alu48$ ./ejemplo1_
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80
Segmentation fault: 11
Tonga51:Desktop fp03alu48$ gcc -g ejemplo1.c -o ejemplo1
Tonga51:Desktop fp03alu48$ gdb ./ejemplo1_
GNU gdb (GDB) 8.2
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-apple-darwin18.2.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.
```

```
Last login: Mon Oct 14 09:29:16 on ttys001
[Tonga51:~ fp03alu48$ servidor
[Tonga51:~ fp03alu48$ ssh fp03alu48@192.168.2.200
The authenticity of host '192.168.2.200 (192.168.2.200)' can't be established
RSA key fingerprint is SHA256:jTgFsbnvP7IaIpwchV27DaUa9i2pvAVVZwZzbIneOF8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.200' (RSA) to the list of known hosts.
[fp03alu48@192.168.2.200's password:
```

```
-bash: aviso: setlocale: LC_CTYPE: no se puede cambiar el local (UTF-8)
[[fp03alu48@samba ~]$ ls
Escritorio
[[fp03alu48@samba ~]$ cd Escritorio
[[fp03alu48@samba Escritorio]$ ls
Proyector.desktop  ejemplo1.c
[[fp03alu48@samba Escritorio]$ gcc -std=c99 -g ejemplo1.c -o ejemplo1
[fp03alu48@samba Escritorio]$
```

gdb

```
[[fp03alu48@samba Escritorio]$ gcc -std=c99 -g ejemplo1.c -o ejemplo1
[[fp03alu48@samba Escritorio]$ gdb ./ejemplo1
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/fp03/fp03alu48/Escritorio/ejemplo1...done.
```

run

```
(gdb) run
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

Program received signal SIGSEGV, Segmentation fault.
0x000000000040060c in main (argc=19, argv=0x1100000010) at ejemplo1.c:21
21                      lista[i] = i;
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
```

list (l)

```
(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)
```

quit (q)

```
        Inferior 1 [process 21650] will be killed.

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

CTRL+X+A

```
[ No Source Available ]

exec No process In:                                          Line: ??   PC: ??
(gdb)
```

start

```
ejemplo1.c
1       #include <stdio.h>
2
3       int main(int argc, char * argv[]) {
4
5               // Asignamos variables
B+>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

child process 22198 In: main                          Line: 6    PC: 0x400542
(gdb) start
Temporary breakpoint 1 at 0x400542: file ejemplo1.c, line 6.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb)
```

next (n)

```
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
9   11          // 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              }
Un numero real: 89.80 un caracter: B
child process 22198 In: main                          Line: 20   PC: 0x40060f
(gdb) start
Temporary breakpoint 1 at 0x400542: file ejemplo1.c, line 6.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb)
```

print (p)

```
12
13              // Mostramos texto y valores
3   14          printf("Primero texto solo\n");
15              printf("Luego podemos poner un entero: %i\n", numero);
5   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);
19  30          printf("El promedio es: %lf\n",promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                          Line: 21   PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1

Temporary breakpoint 1, main (argc=1, argv=0x7fffffffe378) at ejemplo1.c:6
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) p i
$1 = 10 i
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb)
```

display

```
3   14              printf("Primero texto solo\n");
    15              printf("Luego podemos poner un entero: %i\n", numero);
5   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
  >                 // 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);
19  30              printf("El promedio es: %lf\n",promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                                    Line: 21   PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i
display i
1: i = 10
(gdb) display lista
display lista
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb)
```

```
    20              for(int i = numero ; i >= numero ; i++){
  >  21                      lista[i] = i;
  >  22              }
  >  23
  >                 // Y ahora podemos hacer calculos con la lista
    25              for(int i = numero ; i >= numero ; i++){
    26                      suma += lista[i];
  >  20              for(int i = numero ; i >= numero ; i++){
    20              for(int i = numero ; i >= numero ; i++){
  >  20              for(int i = numero ; i >= numero ; i++){
    21                      lista[i] = i;s: %lf\n",promedio);
Un numero real: 89.80 un caracter: B
child process 22945 In: main                                    Line: 21   PC: 0x4005ff
The program is not being run.
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i                                                  0          60
display lista                                                   1          5f
(gdb)                                                           0          60
1: i = 10
(gdb) n
n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 11
(gdb) n
n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 11
(gdb)
```

```
   20              for(int i = numero ; i >= numero ; i++){
>  20              for(int i = numero ; i >= numero ; i++){
   20              for(int i = numero ; i >= numero ; i++){;
>n20              for(int i = numero ; i >= numero ; i++){
 120              for(int i = numero ; i >= numero ; i++){          Line: 21   PC: 0x4005ff
>  20              for(int i = numero ; i >= numero ; i++){
  r21                       lista[i] = i;scritorio/ejemplo1
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb) print lista
print lista
$2 = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
(gdb) display i                                                          0          60
display lista                                                            1          5f
(gdb)                                                                    0          60
1: i = 10                                                                1          5f
1: i = 11                                                                0          60
(gdb) n                                                                  1          5f
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}  0       60
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 12
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 13
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7568, 32767, -7552, 32767}
1: i = 13
(gdb)
```

break (b)

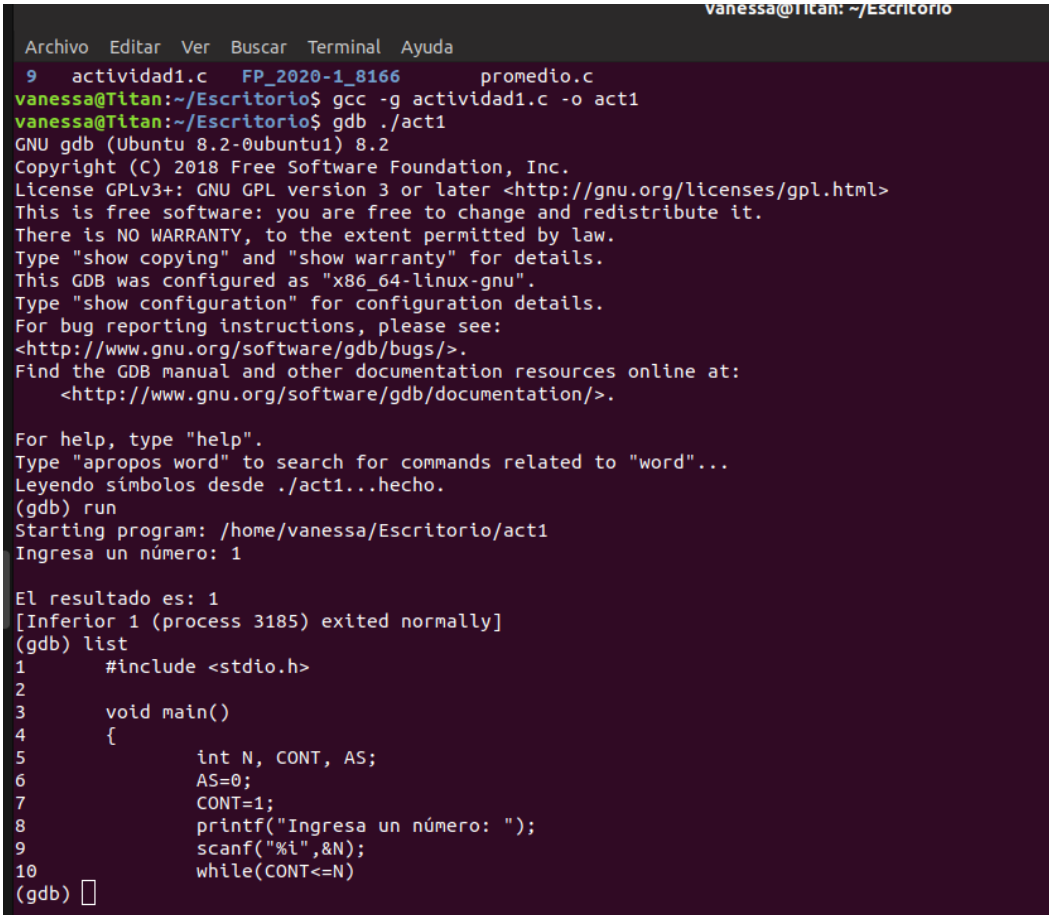
```
(gdb) break 20
Breakpoint 1 at 0x4005f7: file ejemplo1.c, line 20.
(gdb) run
Starting program: /users/fp03/fp03alu48/Escritorio/ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.80

Breakpoint 1, main (argc=1, argv=0x7fffffffe378) at ejemplo1.c:20
20              for(int i = numero ; i >= numero ; i++){
Missing separate debuginfos, use: debuginfo-install glibc-2.15-37.fc17.x86_64
(gdb)
```

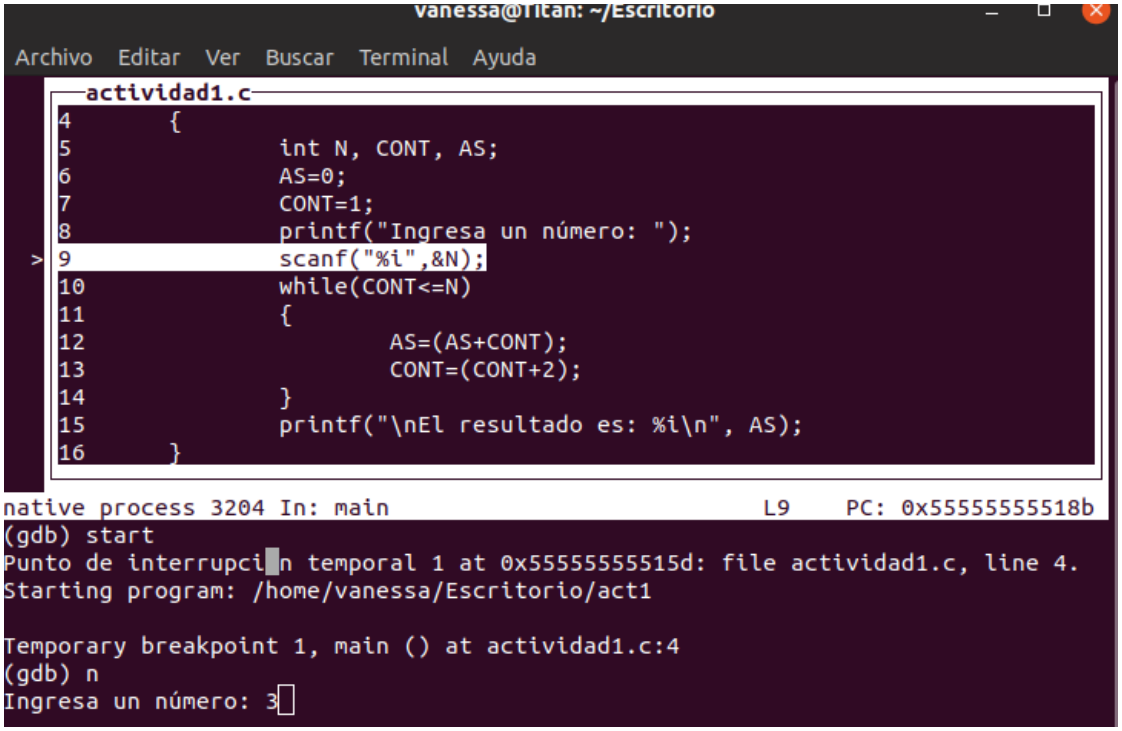
Actividad 1:

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

Para empezar a ver cómo funcionaba, lo compilamos de manera que pudiéramos utilizar gdb, como lo hicimos anteriormente y así ver con detalle su ejecución.

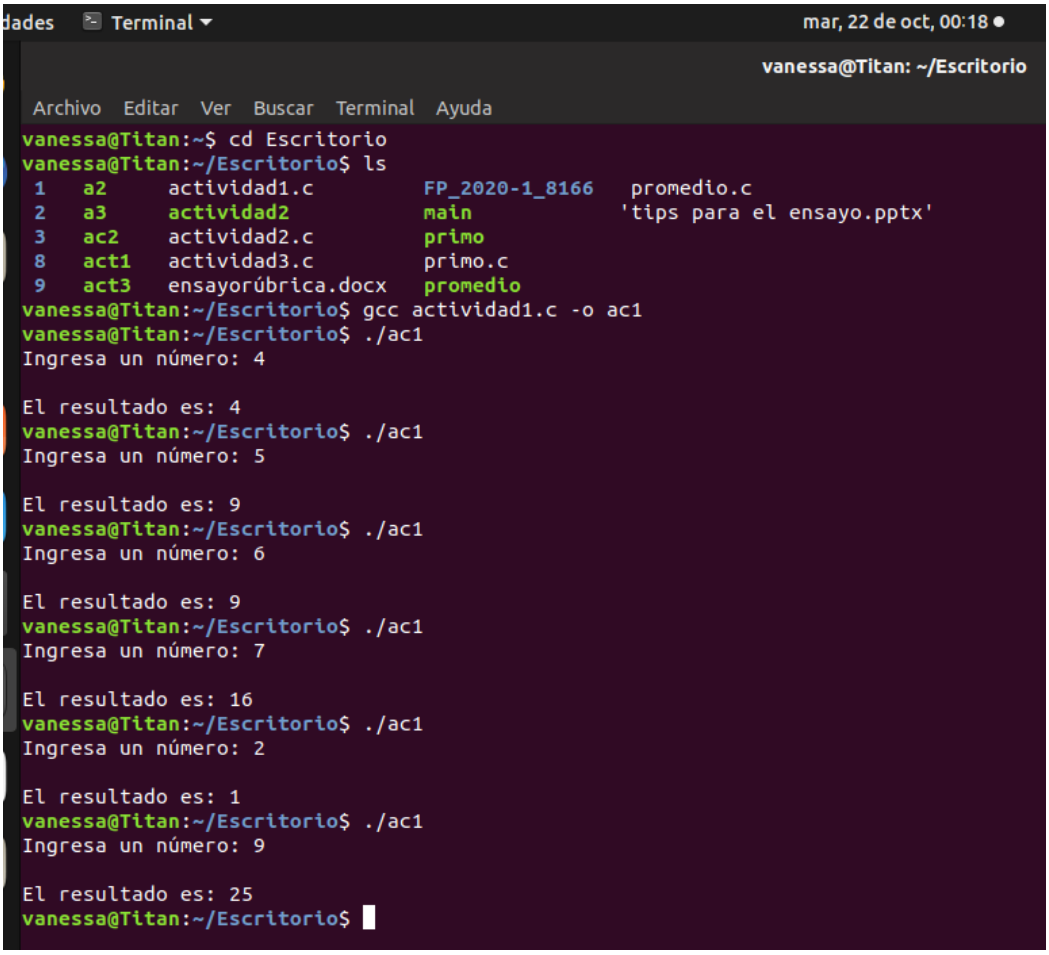


Y después corrimos el programa igual con GDB para poder analizar con mayor precisión.



Después de que analizamos con gdb, al ir saltando renglón por renglón, se ve que la funcionalidad del programa es recibir un número, después en otra variable llamada *AS* se va guardando el nuevo resultado de la **suma** de *AS* más el *contador* que igual es una variable que aumentará de dos en dos e inicia desde 1 y, ésto se repite por un ciclo que tiene como condición que, *contador* aumentará 2 unidades siempre y cuando sea **menor o igual** al valor que se introduce al inicio, por tanto, hasta que *contador* alcance un valor que no sobrepase o sea igual al numero que se introduce, seguirá haciendo los acumulados de la suma tanto en *AS* como en *contador*. Cuando lo alcance, entonces ahí se detiene el ciclo y muestra el resultado total que se fue acumulando en *AS*.

Este fue el resultado después de correrlo en la terminal.



Actividad 2:

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

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

En esta actividad vamos a seguir los mismos pasos que en la primera, solo que con diferente compilación ya que con la primera opción, este programa no se puede compilar.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
1876    in vfscanf.c
vanessa@Titan:~/Escritorio$ gcc -g -w actividad2.c -o a2 -lm
vanessa@Titan:~/Escritorio$ gdb ./a2
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
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-linux-gnu".
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"...
Leyendo símbolos desde ./a2...hecho.
(gdb) run
Starting program: /home/vanessa/Escritorio/a2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=4
X=5
Resultado=6.537500e+01[Inferior 1 (process 2427) exited normally]
(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)
```

```
actividad2.c
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);
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=%le",AS);
23      }
24
25
26
native process 2380 In: main
    errp=errp@entry=0x0) at vfscanf.c:1881
(gdb) start
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Punto de interrupción temporal 2 at 0x55555555515d: file actividad2.c, line 8.
Starting program: /home/vanessa/Escritorio/ac2

Temporary breakpoint 2, main () at actividad2.c:8
(gdb) n
(gdb) n
(gdb) n
N=
```

Y cuando lo corregimos y lo corrimos en la terminal, estos fueron nuestros resultados:

```c
#include <stdio.h>
#include <math.h>

void main()
{
    int K, AP, N;
    double X, AS;
    printf("Ingrese cuántos términos calcular de la serie: X^K/K!");
    printf("\nN=");
    scanf("%i",&N);
    printf("X=");
    scanf("%lf",&X);
    K=0;
    AP=1;
    AS=0;
    while(K<=N)
    {
        AS=AS+pow(X,K)/AP;
        K=K+1;
        AP=AP*K;
    }
    printf("Resultado=%le\n",AS);
}
```

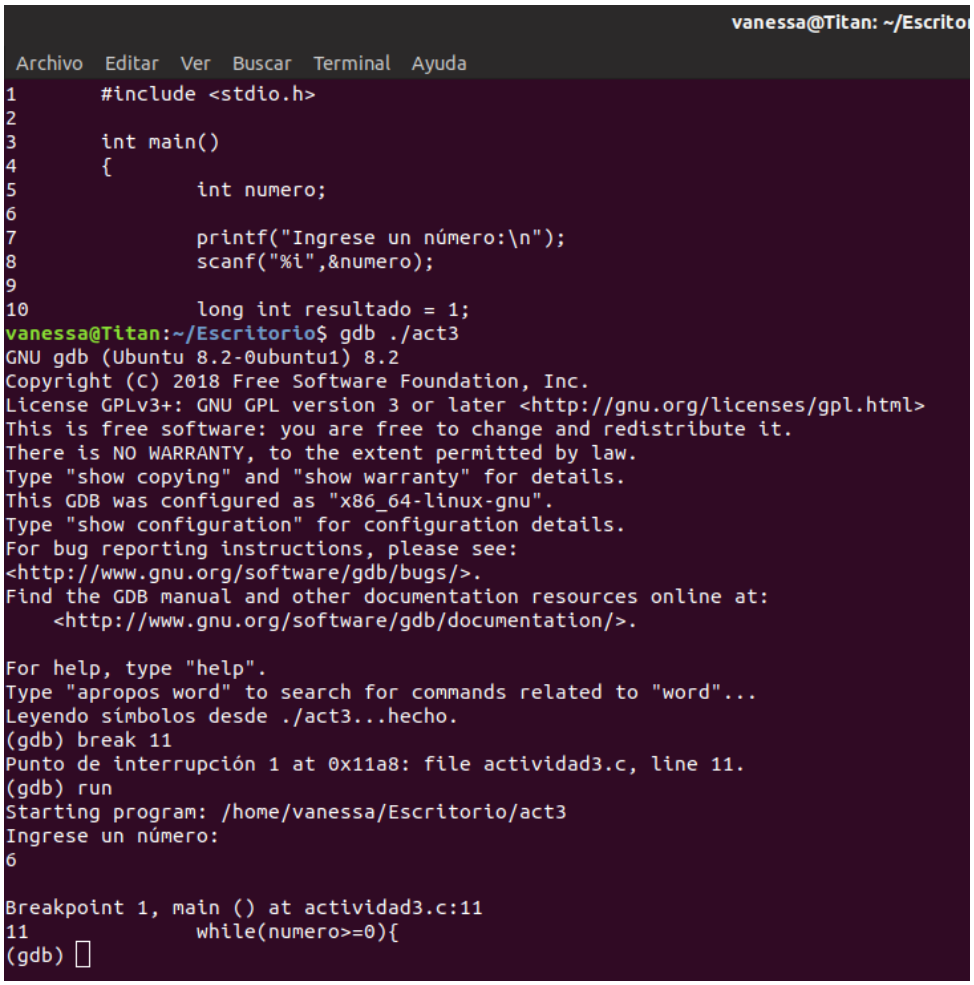
```
vanessa@Titan:~$ cd Escritorio
vanessa@Titan:~/Escritorio$ gcc -w actividad2.c -o acti2 -lm
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=3
X=4
Resultado=2.366667e+01
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=32
X=25
Resultado=-2.224746e+35
vanessa@Titan:~/Escritorio$ ./acti2
Ingrese cuántos términos calcular de la serie: X^K/K!
N=-8
X=-9
Resultado=0.000000e+00
vanessa@Titan:~/Escritorio$
```

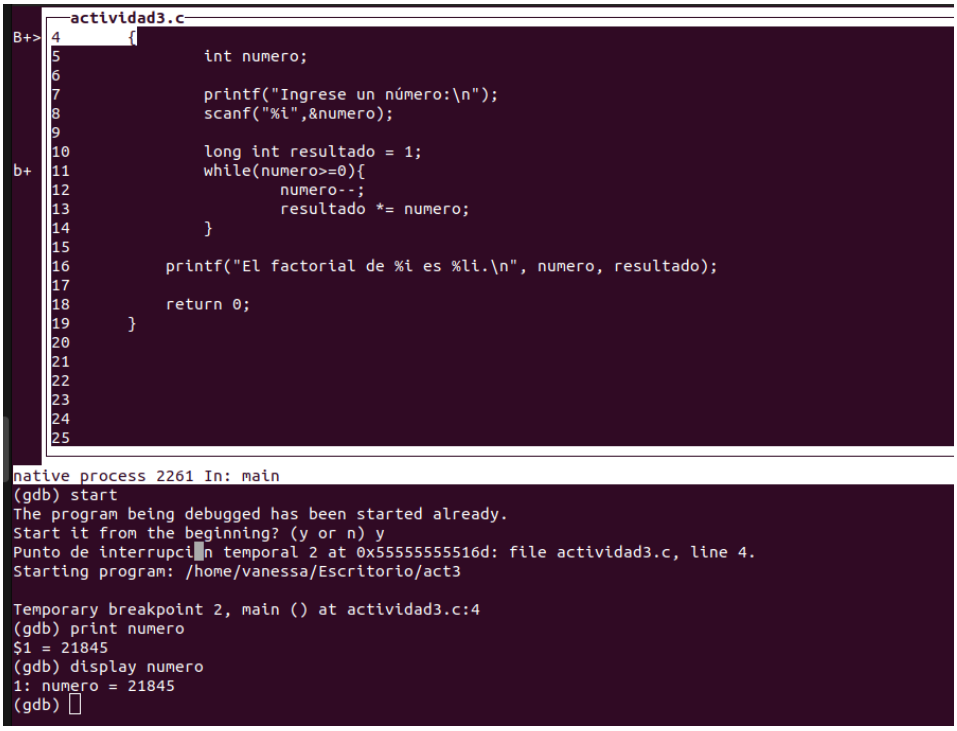
Actividad 3:

- Utilizar GDB para corregir el programa.

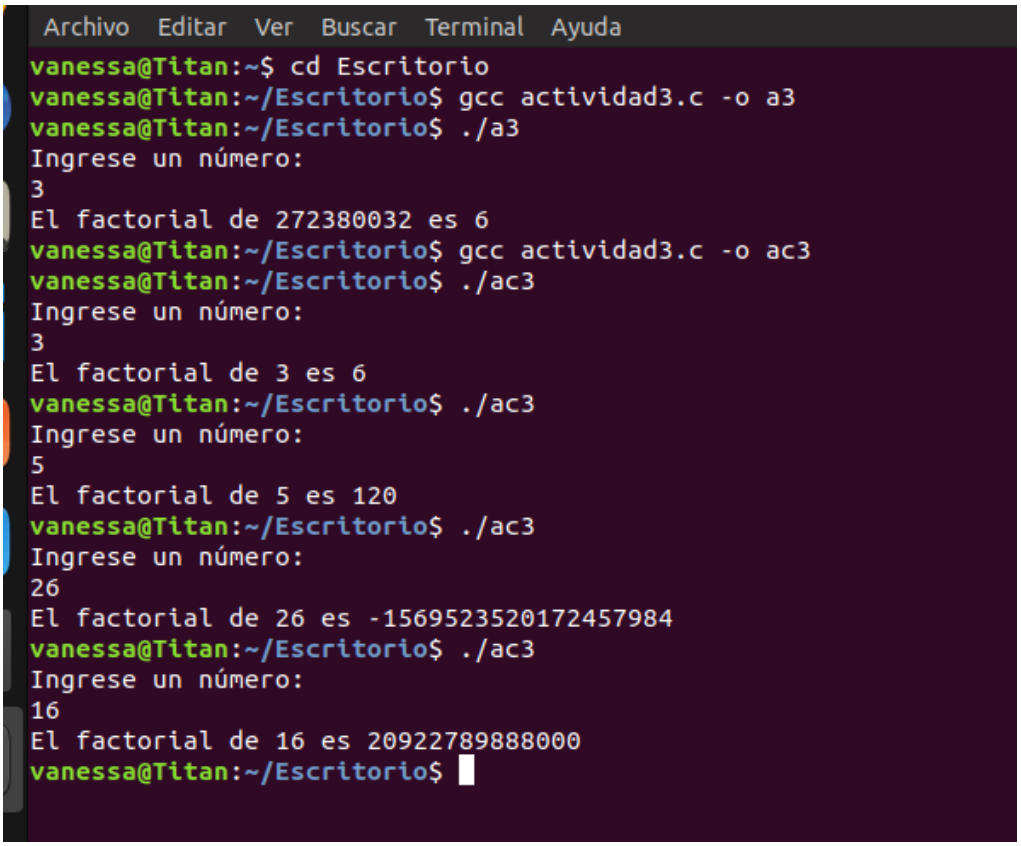
```
vanessa@Titan:~/Escritorio$ gcc actividad3.c -o a3
vanessa@Titan:~/Escritorio$ ./a3
Ingrese un número:
8
El factorial de -1 es 0.
vanessa@Titan:~/Escritorio$ gcc -g actividad3.c -o act3
vanessa@Titan:~/Escritorio$ gdb ./act3
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
Copyright (C) 2018 Free Software Foundation, 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-linux-gnu".
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"...
Leyendo símbolos desde ./act3...hecho.
(gdb) run
Starting program: /home/vanessa/Escritorio/act3
Ingrese un número:
8
El factorial de -1 es 0.
[Inferior 1 (process 2237) exited normally]
(gdb) list
1       #include <stdio.h>
2
3       int main()
4       {
5               int numero;
6
7               printf("Ingrese un número:\n");
```


vanessa@Titan: ~/Escritor
Archivo Editar Ver Buscar Terminal Ayuda
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;
vanessa@Titan:~/Escritorio$ gdb ./act3
GNU gdb (Ubuntu 8.2-0ubuntu1) 8.2
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-linux-gnu".
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"...
Leyendo símbolos desde ./act3...hecho.
(gdb) break 11
Punto de interrupción 1 at 0x11a8: file actividad3.c, line 11.
(gdb) run
Starting program: /home/vanessa/Escritorio/act3
Ingrese un número:
6
Breakpoint 1, main () at actividad3.c:11
11 while(numero>=0){
(gdb)


actividad3.c
B+> 4 {
5 int numero;
6
7 printf("Ingrese un número:\n");
8 scanf("%i",&numero);
9
10 long int resultado = 1;
b+ 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
23
24
25
native process 2261 In: main
(gdb) start
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Punto de interrupción temporal 2 at 0x55555555516d: file actividad3.c, line 4.
Starting program: /home/vanessa/Escritorio/act3
Temporary breakpoint 2, main () at actividad3.c:4
(gdb) print numero
$1 = 21845
(gdb) display numero
1: numero = 21845
(gdb)

Después de corregir algunos errores, estos fueron nuestros resultados:



Conclusión:

Es importante la depuración de programas a nivel profesional, un programa se puede probar con casos específicos del usuario, sin embargo, puede que exista un fallo para una situación en particular que no previó el usuario, para esto sirve la depuración del programa que nos permite probar un programa para cualquier caso posible.

También nos permite encontrar el fallo dentro del código a través de varias funciones, nos permite imprimir funciones y mostrar un listado de una parte del código fuente, nos ayuda a probar casos específicos dentro de la terminal y nos permite saltar líneas de código para el funcionamiento correcto de un programa.