



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Alejandro Pimentel.

*Asignatura:* Fundamentos de programación.

*Grupo:* 3

*No de Práctica(s):* 10

*Integrante(s):* González Segura Areli.  
Rodríguez Guzmán Paola Mariel.  
Villanueva Bustamante

*No. de Equipo de  
cómputo empleado:* 18,19, 20

*No. de Lista o Brigada:* 5319  
4926  
8034

*Semestre:* primer semestre. 2020-1

*Fecha de entrega:* 14 de octubre del 2019

*Observaciones:* Muy bien

**CALIFICACIÓN:** 10

## 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.

## ACTIVIDADES

```
fp03alu44 — fp03alu44@samba:~ — ssh fp03alu44@192.168.2.200 — 80x24
Last login: Mon Oct 14 09:28:24 on ttys000
Mexico20:~ fp03alu44$ servidor
Mexico20:~ fp03alu44$ ssh fp03alu44@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:jTgFsbvP7IaIpachV27DaUa9i2pvAVVZwZzbIneOF8.
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.
fp03alu44@192.168.2.200's password:

Samba

-bash: aviso: setlocale: LC_CTYPE: no se puede cambiar el local (UTF-8)
[fp03alu44@samba ~]$ ls
Escritorio  ejemplo1.c
[fp03alu44@samba ~]$ ls
Escritorio
[fp03alu44@samba ~]$ ls
```

Hubo algunas dificultades para usar la terminal de MAC ya que los códigos que ejecuta bamos no eran correctos, pero pudimos solucionarlos usando SAMBA y así pudimos realizarlo de manera correcta.

Para el primer ejemplo, usamos gdb para ver ue errores podría tener nuestro código, simplemente nos marcaba el error no nos decía que era, así podríamos facilitar mucho más nuestro trabajo. Y se inicia con run.

```
fp03alu44 — fp03alu44@samba:~ — ssh fp03alu44@192.168.2.200 — 80x24
[fp03alu44@samba ~]$ ls
Escritorio  ejemplo1.c
[fp03alu44@samba ~]$ gcc -std=c99
gcc: error fatal: no hay ficheros de entrada
compilaci?n terminada.
[fp03alu44@samba ~]$ gcc -std=c99 -g ejemplo1.c -o ejemplo1
[fp03alu44@samba ~]$ 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/fp03alu44/ejemplo1...done.
(gdb) run
Starting program: /users/fp03/fp03alu44/ejemplo1
Primero texto solo
Luego podemos poner un entero: 10
También podemos poner un caracter: B
Un numero real: 89.88
```

Con el comando list se muestra una línea de código, donde podemos ver todo lo que hicimos. "quit" se usa para salir, y te pregunta si estas seguro de salir.

```
fp03alu44 — fp03alu44@samba:~ — ssh fp03alu44@192.168.2.200 — 80x24

Program received signal SIGSEGV, Segmentation fault.
0x0000000040060c 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
(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
A debugging session is active.

        Inferior 1 [process 21524] will be killed.

Quit anyway? (y or n) y
[fp03alu44@samba ~]$ gdb ejemplo1
GNU gdb (GDB) Fedora (7.4.50.20120120-42.fc17)
```

Y después corrimos otra vez, para usar ctrl+x+a para una interfaz gráfica.

```
fp03alu44 — fp03alu44@samba:~ — ssh fp03alu44@192.168.2.200 — 80x24

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/fp03alu44/ejemplo1...done.
[fp03alu44@samba ~]$ 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/fp03alu44/ejemplo1...done.
(gdb) break 20
Breakpoint 1 at 0x4005f7: file ejemplo1.c, line 20.
(gdb) run
```

```
fp03alu44 — fp03alu44@samba:~ — ssh fp03alu44@192.168.2.200 — 80x24
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
B+> 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     }

child process 22667 In: main                                Line: 20    PC: 0x4005f7
(gdb)
```

para pasar a la siguiente línea se usaba el comando “n”, con “p” se puede imprimir una variable, también se puede dar seguimiento a las variables con el comando “display”

```
ejemplo1.c
19     // Podemos llenar la lista con valores
B+> 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

child process 22667 In: main                                Line: 21    PC: 0x4005ff
1: i = 12
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7536, 32767, -7520, 32767}
1: i = 12
(gdb) n
2: lista = {-163754450, 0, 4195102, 0, -1, 0, -7536, 32767, -7520, 32767}
1: i = 13
(gdb) n
```

## Actividad:

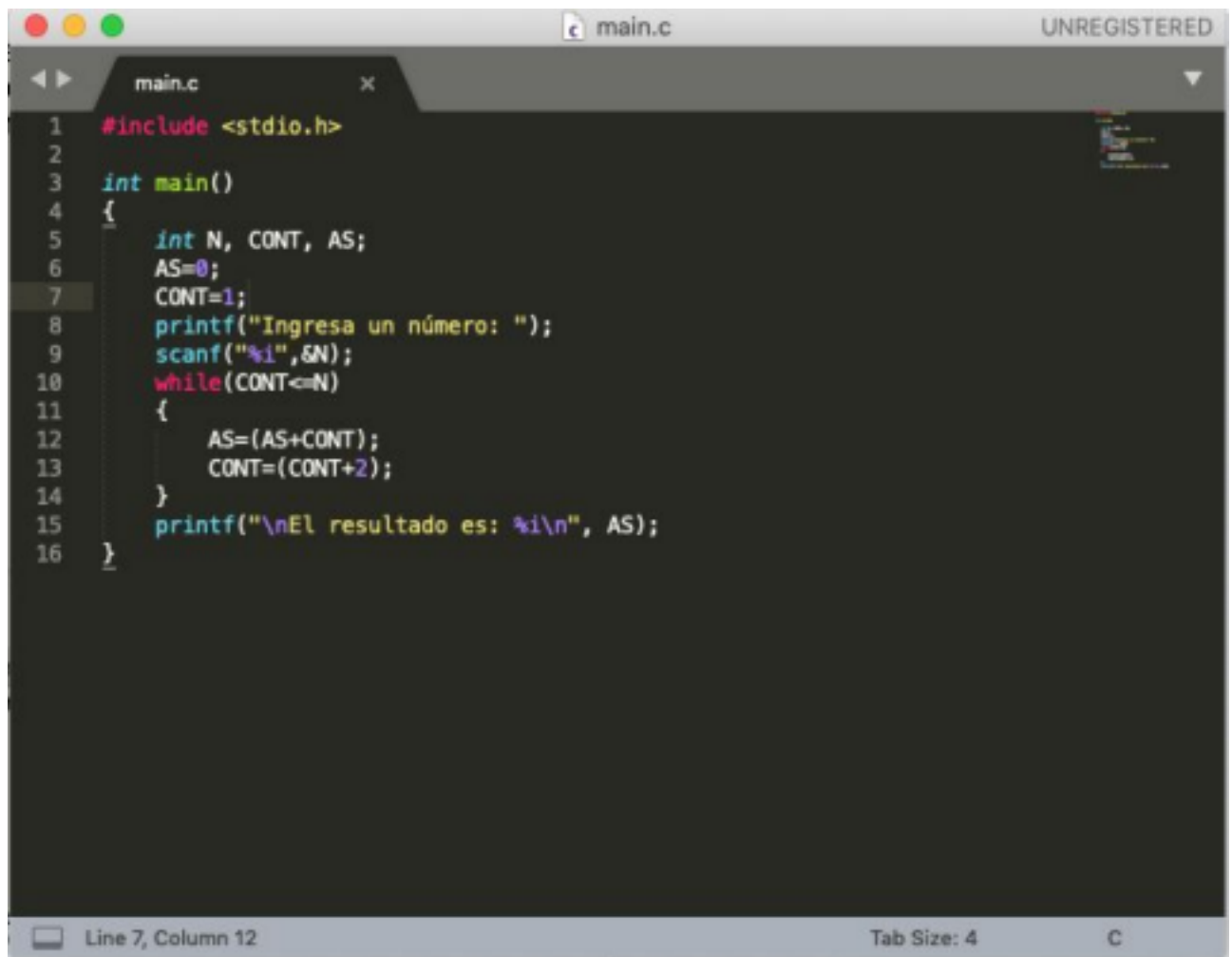
### 1. Utilizar GDB para encontrar la utilidad del programa y describir su funcionalidad.

Tiene como objetivo el permitirle al usuario ver lo que esta ocurriendo en su programa durante su ejecución o lo que ocurrió antes de que el programa parara de ejecutar a razón de un error (bug).

Es una herramienta fundamental en el desarrollo y corrección de fallas. El gdb puede ser utilizado para depurar fallas en varios lenguajes incluyendo C.

El gdb es un depurador de texto, todos sus comandos son instrucciones textuales.

Una vez que se reviso el programa se noto que el error estaba en el "void main()" y se cambio por un int main()



```
1  #include <stdio.h>
2
3  int 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)
11     {
12         AS=(AS+CONT);
13         CONT=(CONT+2);
14     }
15     printf("\nEl resultado es: %i\n", AS);
16 }
```

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 cont que igual es una variable que aumentará de dos en dos , y esto se repite por un ciclo que tiene como condición que el contador sea menor o igual al número ingresado. Al compilar nos sale esto como resultado.



```
Documents — -bash — 80x24
Last login: Mon Oct 28 09:06:56 on ttys000
Mexico20:~ fp03alu44$ cd
.CFUserTextEncoding  Documents/      Pictures/
.Trash/              Downloads/    Public/
.bash_history        Library/     act.c
.bash_sessions/     Movies/
Desktop/            Music/
Mexico20:~ fp03alu44$ cd Documents/
Mexico20:Documents fp03alu44$ gcc act.c -o act
act.c:3:1: warning: return type of 'main' is not 'int' [-Wmain-return-type]
void main()
^
act.c:3:1: note: change return type to 'int'
void main()
^~~~
int
1 warning generated.
Mexico20:Documents fp03alu44$ gcc main.c -o main
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 4

El resultado es: 4
Mexico20:Documents fp03alu44$
```

```
Documents — -bash — 80x24
int
1 warning generated.
Mexico20:Documents fp03alu44$ gcc main.c -o main
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 4

El resultado es: 4
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 6

El resultado es: 9
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 7

El resultado es: 16
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 19

El resultado es: 100
Mexico20:Documents fp03alu44$ ./main
Ingresa un número: 12

El resultado es: 36
Mexico20:Documents fp03alu44$
```

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

Se pusieron notas con lo que se corrigió:

```

actividad2.c
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 cuantos t rminos calcular de la serie: X^K/K!\n");
9      printf("\nN=");
10     //falt  la & en la N del scanf
11     scanf("%i", &N);
12     printf("X=");
13     //falt  la & en la X del scanf
14     scanf("%lf", &X);
15     K=0;
16     AP=1;
17     AS=0;
18     //se cambia solo por un < para que se haga el proceso
19     while(K<N)
20     {
21         AS=AS+pow(X,K)/AP;
22         K=K+1;
23         AP=AP*K;
24     }
25     printf("Resultado=%le", AS);
26 }

```

Y as  ya sale el resultado

```

C:\Users\ANAH\Downloads\actividad2.exe
Ingrese cuantos t rminos calcular de la serie: X^K/K!
N=3
X=4
Resultado=1.300000e+001
-----
Process exited after 14.67 seconds with return value 23
Presione una tecla para continuar . . .

```

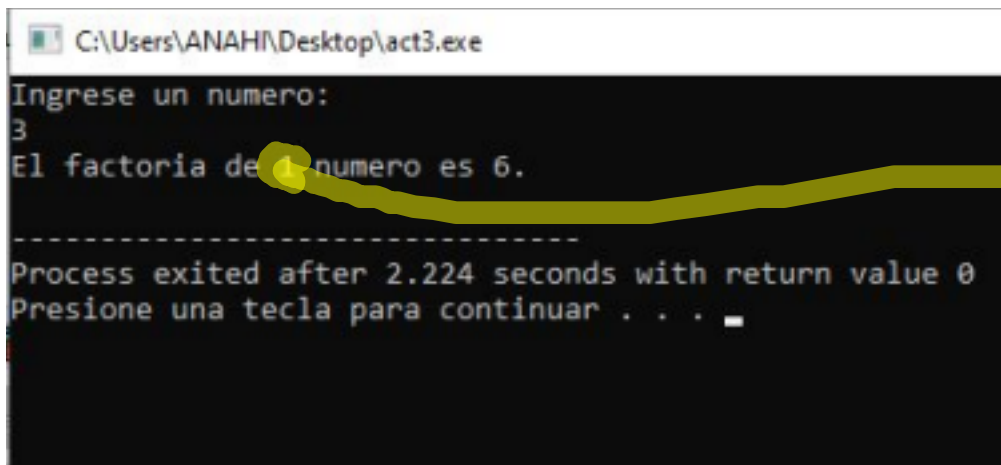
### 3. Utilizar GDB para corregir el programa.

Se pusieron lo que se cambi  en un comentario

```

act3.c
3  int main()
4  {
5      long int numero;
6      //a numero tambi n se le agrega long para que fuyan los mismos resultados
7      //se agrega otra variable para el resultado
8      //Se declaro la variable "resultado primero"
9      long int resultado=1;
10
11
12     printf("Ingrese un numero: \n");
13     scanf("%i", &numero);
14
15     resultado = 1;
16     //se ayuso que el numero fuera mayor a 1 no siendo el igual
17     while(numero>1){
18
19         resultado *= numero;
20         //El numero-- se puso despues para que afectara una vez que tuviera el resultado
21         numero--;
22     }
23
24     printf("El factoria de %i numero es %li.\n", numero, resultado);
25
26     return 0;
27 }
28

```



```
C:\Users\ANAH\Desktop\act3.exe
Ingrese un numero:
3
El factoria de 3 numero es 6.
-----
Process exited after 2.224 seconds with return value 0
Presione una tecla para continuar . . . _
```

Cuidado aquí

### **conclusión:**

Usar gdb es muy útil para ver errores en tu programa y cambiarlos y así funcione correctamente, a la hora de compilarlo, por eso fue muy bueno saber con esta práctica su funcionalidad y la manera de usarlo.