USO DEL DEPURADOR GBD

TEMA:	DEPURACIÓN Y ANÁLISIS DE PROGRAMAS

INDICACIONES GENERALES

- Revisar este material antes del desarrollo de la sesión de laboratorio correspondiente.
- Para un mejor desempeño, complementar el estudio de esta guía con el material del curso y la bibliografía indicada.
- Revisar y desarrollar los ejemplos por su cuenta y comparar su solución con la presentada en esta guía.

OBJETIVOS .

- Manejar la herramienta de depuración gdb.
- Realizar seguimiento de las variables locales, registros, y posiciones de memoria de interés de un programa.

ÍNDICE

DEPURACIÓN

Depurar un programa con fallas (**debug**) es el proceso de confirmar, sentencia por sentencia, que los supuestos que afirmamos que hace el programa se cumplen. Cuando se ubica la línea exacta de código en la cual esos supuestos no se cumplen, es cuando se afirma que se ha detectado la falla (**bug**).

Depurador de GNU (GDB)

GDB es la herramienta de depuración más empleada por los usuarios de distribuciones UNIX/LINUX. Por lo general, GDB viene instalado por defecto, pero en caso no sea así lo puede instalar ejecutando los siguientes comandos en un terminal:

```
$ sudo apt-get update
$ sudo apt-get install gdb
```

Manejo del GDB

Se hará una presentación general de los comandos que nos servirán para hacer un seguimiento del programa con el siguiente ejemplo codificado en 64 bits:

```
1  ; para ensamblar:
2  ; nasm -f elf64 -g ejemplo_gdb.asm -o ejemplo_gdb.o
3  ; para enlazar:
4  ; ld ejemplo_gdb.o -o ejemplo_gdb
5          segment .data
6  a       dd      4
7  b       dd      4.4
8  c       dw      1,2
9  e       db      "hola mundo",10,0
10 f       equ     $-e
11
12         segment .bss
13 g       resb    1
14 h       resw    10
15 i       resq    100
16
17         segment .text
18         global _start
19
20 _start:
21
22         mov     rax, 1
23         mov     rdi, 1
24         mov     rsi, e
25         mov     rdx, f
26         syscall
27
28         mov     rax, 60
```

```
29          mov     rdi, 0
30          syscall
```

Comandos generales de GDB

Antes de ejecutar el depurador, debe estar seguro de que ha colocado la bandera **-g** en el momento de la compilación. A pesar de que se menciona en la cabecera del archivo se le recuerda que debe ser de la siguiente manera:

```
$ nasm -f elf64 -g ejemplo_gdb.asm -o ejemplo_gdb.o
$ ld ejemplo_gdb.o ejemplo_gdb
```

NOTA: Si el depurador requiere utilizar en programas codificados en 32 bits, la forma de generar el objeto debería ser:

```
$ nasm -f elf32 -g ejemplo_gdb.asm -o ejemplo_gdb.o
$ ld -m elf_i386 ejemplo_gdb.o -o ejemplo_gdb
```

Para correr el depurador con nuestro programa ejemplo deberá ejecutar el siguiente comando:

```
$ gdb ejemplo_gdb
```

Luego de ejecutar el comando deberá tener en el terminal un resultado **similar** al siguiente:

```
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
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"...
Reading symbols from ejemplo_gdb...done.
(gdb)
```

a. **Fijar un punto de quiebre en el código**

```
(gdb) break _start
Breakpoint 1 at 0x4000b0
```

Este comando en general debe indicar la instrucción **break** y la **etiqueta** en la que se desea hacer la pausa. Puede haber más de un **breakpoint**.

b. **Ver la información de los breakpoints**

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004000b0 <_start>
        breakpoint already hit 1 time
```

c. **Borrar un breakpoint**

```
(gdb) delete 1
```

Para este comando basta con indicar el número del breakpoint luego de delete.

d. **Ejecutar el código**

```
(gdb) run
Starting program: /home/stefanos/Documents/asm/ejemplos/gdb/ejemplo_gdb

Breakpoint 1, 0x00000000004000b0 in _start ()
```

Si se han fijado **breakpoints** el programa se ejecutará hasta el primero de estos, en caso contrario ejecutará todo el programa.

e. **Fijar la sintaxis**

```
(gdb) set disassembly-flavor intel
```

Este comando permitirá observar la salida del **disassembly** con la sintaxis que empleamos.

f. **Mostrar el disassembly**

```
(gdb) disassemble
Dump of assembler code for function _start:
   0x00000000004000b0 <+0>:     mov    eax,0x1
=> 0x00000000004000b5 <+5>:     mov    edi,0x1
   0x00000000004000ba <+10>:    movabs rsi,0x6000e5
   0x00000000004000c4 <+20>:    mov    edx,0x6
   0x00000000004000c9 <+25>:    syscall
   0x00000000004000cb <+27>:    mov    eax,0x3c
   0x00000000004000d0 <+32>:    mov    edi,0x0
   0x00000000004000d5 <+37>:    syscall
End of assembler dump.
```

Muestra el código ensamblado y su lugar en la memoria de cada instrucción. La flecha en la parte izquierda indica la instrucción en la que se encuentra y **aún no ha ejecutado**.

g. **Mostrar variables**

```
(gdb) info variables
All defined variables:

Non-debugging symbols:
0x00000000006000d8  a
0x00000000006000dc  b
0x00000000006000e0  c
0x00000000006000e4  d
0x00000000006000e5  e
0x00000000006000f1  __bss_start
```

```
0x00000000006000f1  _edata
0x00000000006000f4  g
0x00000000006000f5  h
0x0000000000600109  i
0x0000000000600430  _end
```

Muestra las variables contenidas el segmento .data y el segmento .bss así como sus posiciones de memoria.

h. **Siguiente instrucción**

```
(gdb) nexti
```

Comandos para seguimiento de registros de GDB

a. **Mostrar la información de los registros de propósito general.**

```
(gdb) info registers
rax            0x1      1
rbx            0x0      0
rcx            0x0      0
rdx            0x0      0
rsi            0x0      0
rdi            0x0      0
rbp            0x0      0x0
rsp            0x7fffffffde20   0x7fffffffde20
r8             0x0      0
r9             0x0      0
r10            0x0      0
r11            0x0      0
r12            0x0      0
r13            0x0      0
r14            0x0      0
r15            0x0      0
rip            0x4000b5 0x4000b5 <_start+5>
---Type <return> to continue, or q <return> to quit---
```

Muestra los valores contenidos en los registros de propósito general.

b. **Mostrar la información de todos los registros.**

```
(gdb) info all-registers
```

Muestra los valores contenidos en **todos** los registros.

Comandos para mostrar elementos de memoria

El formato para el comando **print** es **print expression** o **print/format expression** donde **format** es una letra que define el formato de las datos que se van a mostrar. Los formatos a elegir pueden ser algunos de los siguientes:

Cuando emplea el comando print sobre una variable es recomendable indicar el tipo de dato adecuado para mostrar el resultado correcto.

```
(gdb) print a
'a' has unknown type; cast it to its declared type
```

letra	formato
d	decimal
x	hexadecimal
t	binario
u	sin signo
f	punto flotante
i	instrucción
c	char
s	string
a	address

```
(gdb) print (char) a
$7 = 4 '\004'
(gdb) print (short) a
$8 = 4
(gdb) print (int) a
$9 = 4
(gdb) print (long) a
$10 = 4651317696007241732
(gdb) print (float) b
$11 = 4.4000001
(gdb) print (double) b
$12 = 2.7813688936711431e-309
(gdb) print/u (int) a
$13 = 4
(gdb) print/x (int) a
$14 = 0x4
```

Comandos para examinar elementos de memoria

El formato para examinar es **x/CantFormTam direccion** donde **Cant** es el número de elementos a examinar, **Form** es el formato, y **Tam** es el tamaño de cada posición de memoria. Los tamaños a elegir pueden ser algunos de los siguientes:

letra	tamaño	bytes
b	byte	1
h	halfword	2
w	word	4
g	giant	8

```
(gdb) x/w &a
0x6000d8 <a>: 0x00000004
```

```
(gdb) x/w 0x00000000006000d8
0x6000d8 <a>: 0x00000004
(gdb) x/fg &b
0x6000dc <b>: 2.7813688936711431e-309
(gdb) x/fw &b
0x6000dc <b>: 4.4000001
(gdb) x/2xh &c
0x6000e0 <c>: 0x0001 0x0002
(gdb) x/10cb &e
0x6000e5 <e>: 104 'h'111 'o'108 'l'97 'a'32 ' '109 'm'117 'u'110 'n'
0x6000ed: 100 'd'111 'o'
(gdb) x/s &e
0x6000e5 <e>: "hola mundo\n"
(gdb) x/s 0x00000000006000e5
0x6000e5 <e>: "hola mundo\n"
```