Autor: Javier Aranguren Ortiz

GUIA BÁSICA GDB

ÍNDICE

Autor: Javier Aranguren Ortiz

CODIGO DE COLORES

Negrita: Código.

Subrayado amarillo: Instrucción.

Subrayado azul: Respuesta a una instrucción.

COMPILAR

Compilar como código objeto:

1) Lenguaje c
   gcc -m32 -g -o sum1toN sum1toN.c
2) Lenguaje ensamblador(Si no está _start, saltara un warning)
   gcc -nostartfiles -m32 -g -o fichero fichero.s

GDB

PRACTICA 1

1) Guardar las instrucciones y sus respuestas en un fichero:
   set trace-commands on
   set logging file fichero.txt
   set logging on

2) Cargar modulo objeto en gdb
   file fichero

3) Comandos básicos gdb
   a. **break** / **b** : Crea un punto de ruptura en el programa.
      b 21
      Punto de interrupción 1 at 0x565561ad: file sum1toN.c, line 22.
   b. **run** / **r** : Ejecuta hasta el primer punto de ruptura, en el caso de que no exista punto de ruptura ejecuta todas las líneas, la línea del punto de ruptura no se ejecuta.
      r
      Starting program: /home/javier/Escritorio/Pruebas eedd/sum1toN
   c. **next** / **n** : Ejecuta la instrucción de la línea y avanza a la siguiente.
      n
   d. **print** : Como se ha explicado antes, esta instrucción permite mostrar permite mostrar información de las variables y direcciones de memoria, tiene varias variantes:
      i. **print expresión** : Muestra el contenido de una expresión.
         p n
         $4 = 0 '\000'
      ii. **print /Formato expresión** : Muestra el contenido una expresión en el formato indicado.

      Formatos:

o – octal	u – decimal sin signo	t – binario	c – carácter
x - hexadecimal	d – decimal	a – dirección	s – cadena

```
p /f sum        $5 = -5
p /t sum        $6 = 11111011
```

e. **continue** / **c** : Continua hasta el siguiente punto de ruptura, si no hay ejecutara las líneas restantes.

```
c
Continuando.
```

f. **info break** / **i b** : Muestra información sobre los puntos de ruptura creados, su utilidad es obtener el "num" para poder borrarlos en caso de necesitarlo.

```
I b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x565561ad in main at sum1toN.c:22
        breakpoint already hit 1 time
2       breakpoint     keep y   0x565561ea in main at sum1toN.c:29
```

g. **delete** / **d** : Elimina todos los brakpoints si no incluye argumentos, en el caso de incluir un argumento, este debe ser el "num" del punto de ruptura que se quiere eliminar.

```
d 1
```

h. **until** / **u**: Avanza hasta la línea especificada, debe haberse ejecutado run antes.

```
u 29
main () at sum1toN.c:29
```

i. **quit** / **q**: Cierra el programa gdb.

```
q
```

PRACTICA 2

1) print de registros: Para poder mostrar el contenido de los registros debo usar $ delante de estos.

```
p $si     $8 = 4
```

2) **x** : Es similar al print, actúa sobre direcciones de memoria.
   a. **x dirección** : muestra el contenido de la dirección.

```
x &lista          0x5655800b:    00000001
```

   b. x /NFT dirección

   Donde:

   -**N**: Número de T que se quieren leer

   -**F**: Formato de salida:

o – octal	u – decimal sin signo	t – binario	c – carácter
x - hexadecimal	d – decimal	a – dirección	s – cadena

   -**T**: Tamaño que se quiere leer

b – byte(1 byte)	h – halfword (2 byte)	w – Word(4 byte)	g – giant(8 byte)

```
x /5xb &men1          0x68    0x6f    0x6c    0x61    0x01
```

3) **(tipo)** : Casteo: Transforma el tipo de dato.

```
x /dw (int *)&lista+1     2
```

PRACTICA 4

1) Imprime los nombres de los flags activos.

```
p $eflags
```

2) Interrumpe la ejecución y visualiza el contenido del registro EFLAGS ←- cada vez que cambia su valor.
   watch $eflags
3) Visualiza los watchs definidos
   info watch
4) Elimina los breaks, watchs, etc
   delete breakpoints

PRACTICA 6

1) Top del stack: x $sp : stack pointer
2) Bottom del frame: x $fp : frame pointer