

GLOSARIO GDB

Presentamos un glosario que creemos que les será útil para aprender a utilizar GDB. Para más información vea la sección “**Un poco de GDB**”.

GDB es una herramienta que permite debugear y les puede ser de mucha ayuda en momentos críticos. Una vez que sepan utilizarlo, codear/programar puede volverse una tarea muy ágil. Durante toda la primera parte de la materia irán encontrándose con ejercicios complejos que necesitarán un seguimiento muy (**muy**) de cerca. Por esta razón, encontrar un error dentro del código puede tomar 2 horas o 5 minutos dependiendo de qué tan buen uso le den a debugger.

En principio puede parecerles difícil de entender, o incluso puede asustar un poquito la interfaz que tiene, pero confíen en que cuánto más uso le den, la navegación será más amena y llegará un punto donde no podrán programar sin utilizar esta herramienta. A continuación les brindamos los comandos que según nuestro criterio son los más utilizados y que necesitarán utilizar mucho. Los comandos deben escribirse en la terminal con el formato que indicamos.

COMANDOS

gdb	<p>Nos permite abrir el debugger.</p> <p>Para esto debemos estar posicionados en un directorio que contenga el archivo que queramos correr.</p> <p>Cuando lo ejecutamos veremos algo de la siguiente pinta:</p> <pre>GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git Copyright (C) 2024 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: <https://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". >>> █</pre> <p>Con este comando no indicamos el archivo a debugear. Se necesita hacer un paso más (véase run file).</p>
gdb file	<p><u>Es el primer paso para comenzar a trabajar.</u></p> <p>Nos permite debugear el ejecutable llamado <i>file</i>.</p> <p>Por ejemplo, si deseamos debugear los tests, y el ejecutable se llama <i>tester</i>, debemos escribir “<i>gdb tester</i>”. De este modo le indicamos a la herramienta el archivo que deseamos que cargue.</p>

run	<p>Comando equivalente: r. Nos permite correr el programa. Si no le indicamos a la herramienta previamente qué programa ejecutar, saldrá un aviso del estilo:</p> <pre>>>> r Starting program: No se especificó un archivo ejecutable. Use la orden «file» o «exec-file».</pre> <p>Cuando lo ejecutamos suele requerir que respondamos:</p> <pre>This GDB supports auto-downloading debuginfo from the following URLs: <https://debuginfod.ubuntu.com> Enable debuginfod for this session? (y or [n])</pre> <p>Una vez indicada su respuesta, el programa se ejecutará.</p>
quit	Nos permite cerrar el debugger
b <u>nombre_de_funcion</u>	Nos permite colocar un breakpoint. El breakpoint es para indicar dónde frenar la ejecución. Aquí le pedimos que frene en la función llamada " <u>nombre_de_funcion</u> "
b <u>nombre_de_archivo:numero_de_linea</u>	<p>Nos permite colocar un breakpoint en el archivo "<u>nombre_del_archivo</u>" en la línea "<u>numero_de_linea</u>". (No olvidarse de los dos puntos en el medio.)</p>
info b	Nos permite conocer todos los breakpoints que tenemos colocados en el programa. Nos imprime una lista con la información de cada uno de ellos.
delete <u>número_de_breakpoint</u>	Nos permite eliminar un breakpoint. Cada breakpoint está identificado por un número id (véase info b).
delete breakpoints	Nos permite eliminar todos los breakpoints.
next	<p>Comando similar: n. Nos permite avanzar de línea. No ingresa a las funciones, solo avanza a la siguiente línea.</p>

step	<p>Comando similar: s.</p> <p>Nos permite ingresar a una función.</p>
continue	<p>Comando similar: c.</p> <p>Nos permite avanzar al siguiente breakpoint. Si no hay un breakpoint, llega hasta el final del programa.</p>
print <i>nombre_de_variable</i>	<p>Comando similar: p <i>nombre_de_variable</i>.</p> <p>Nos permite imprimir en consola el contenido de la variable "<u><i>nombre_de_variable</i></u>".</p> <p>Formato: Imprime según el tipo de dato de la variable.</p> <p>Ejemplo: En un ciclo for, si queremos conocer el valor del iterador i, podemos hacer:</p> <pre>>>> p i \$10 = 0 >>></pre>
print \$<i>nombre_de_registro</i>	<p>Comando similar: p \$<i>nombre_de_registro</i>.</p> <p>Nos permite imprimir en consola el contenido del registro "<u><i>nombre_de_registro</i></u>".</p> <p>Formato: decimal.</p> <p>TIP: Utilicen los diferentes tamaños de los registros. Por ejemplo: p \$rdx o también p \$dl.</p>
p/x <i>nombre_de_variable</i> p/x \$<i>nombre_de_registro</i>	<p>Nos permite imprimir en consola el contenido del registro "<u><i>nombre_de_registro</i></u>" o la variable "<u><i>nombre_de_variable</i></u>".</p> <p>Formato: hexadecimal.</p> <p>Ejemplo: Si queremos conocer el valor de la variable i, podemos hacer:</p> <pre>>>> p/x i \$11 = 0x1 >>></pre>
p/u <i>nombre_de_variable</i> p/u \$<i>nombre_de_registro</i>	<p>Nos permite imprimir en consola el contenido del registro "<u><i>nombre_de_registro</i></u>" o la variable "<u><i>nombre_de_variable</i></u>".</p> <p>Formato: decimal unsigned.</p>

<p>p/c nombre_de_variable p/c \$nombre_de_registro</p>	<p>Nos permite imprimir en consola el contenido del registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>". Formato: char. Carácter ASCII.</p>
---	--

<p>p/f nombre_de_variable p/f \$nombre_de_registro</p>	<p>Nos permite imprimir en consola el contenido del registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>". Formato: float.</p>
---	---

<p>p (uint32_t*) nombre_de_variable p (uint32_t*) \$nombre_de_registro</p>	<p>Nos permite imprimir en consola el contenido del registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>". Formato: puntero a uint32_t. <i>Obs: No imprime el contenido en memoria. Solo imprime el puntero (dirección de memoria). Con uint32_t indicamos al tipo de dato al que apunta.</i></p>
---	--

<p>p *(uint32_t*) nombre_de_variable p *(uint32_t*) \$nombre_de_registro</p>	<p>Comando similar: x/(uint32_t) \$nombre_de_registro . Nos permite imprimir en consola la memoria apuntada por el puntero contenido en el registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>". Formato: Lee la memoria como un dato de tipo <u>uint32_t</u>, entonces imprime en ese formato.</p>
---	---

<p>p *(uint32_t*) nombre_de_variable p *(char*) \$nombre_de_registro</p>	<p>Comando similar: x/c \$nombre_de_registro. Nos permite imprimir en consola la memoria apuntada por el puntero contenido en el registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>". Formato: Lee la memoria como un carácter, entonces imprime en ese formato.</p>
---	---

<p>p ({char} \$nombre_de_registro)@numero_de_bytes</p>	<p>Nos permite imprimir una cantidad (numero_de_bytes) de</p>
---	--

	<p>elementos de un tipo determinado ({char}) a partir de una dirección (\$nombre_de_registro).</p> <p><i>TIP: Útil cuando conocemos la dirección y el tipo de una variable y queremos ver su contenido.</i></p> <p><i>TIP2: en lugar de poner \$nombre_de_registro podemos escribir directamente de la dirección de memoria.</i></p>
--	---

x nombre_de_variable x \$nombre_de_registro	<p>Nos permite imprimir la memoria apuntada por el puntero contenido en el registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>".</p>
--	--

Nota: El comando x está explicado en profundidad en la sección "Un poco de GDB".

x/4 nombre_de_variable x/4 \$nombre_de_registro	<p>Nos permite imprimir 4 bytes de memoria apuntada por el puntero contenido en el registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>".</p> <p><i>TIP: Útil para conocer el contenido de un array.</i></p> <p><i>TIP2: En lugar del 4 se coloca el número de bytes deseado e imprime dicha cantidad.</i></p>
--	---

x/4c nombre_de_variable x/4c \$nombre_de_registro	<p>Nos permite imprimir 4 bytes de memoria apuntada por el puntero contenido en el registro "<u>nombre_de_registro</u>" o la variable "<u>nombre_de_variable</u>".</p> <p>Formato: 4 bytes como char.</p> <p><i>TIP: Útil para conocer el contenido de un array.</i></p> <p><i>TIP2: En lugar del 4 se coloca el número de bytes deseado e imprime dicha cantidad.</i></p> <p><i>TIP3: En lugar de c se coloca el tipo de dato deseado.</i></p>
--	--