Jose David Tello Medina - 1216721757

Laboratorio de memoria

1. Sucede un error de violación de segmento, la cual no deja ejecutar correctamente el programa.

2. el debugger gdb muestra un fallo de segmentación en el lugar donde se asigna el nuevo valor del apuntador, en este caso es la línea 7

```
└─→ $gdb null.out
GNU gdb (Debian 9.1-2) 9.1
Copyright (C) 2020 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 null.out...
(gdb) r
Starting program: /home/ft/Desktop/carpeta sin título/memory-api/P1/null.out

Program received signal SIGSEGV, Segmentation fault.
0x0000555555555144 in main () at null.c:7
7           *p = 30;
(gdb)
```

3. En este caso, valgrind muestra algo similar a gdb, en el cual especifica que la dirección de memoria no fue apilada y por lo tanto no se puede realizar esa operación.

```
└─→ $valgrind --leak-check=yes ./null.out
==43165== Memcheck, a memory error detector
==43165== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==43165== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==43165== Command: ./null.out
==43165==
==43165== Invalid write of size 4
==43165==    at 0x109144: main (null.c:7)
==43165==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==43165==
==43165==
==43165== Process terminating with default action of signal 11 (SIGSEGV)
==43165==  Access not within mapped region at address 0x0
==43165==    at 0x109144: main (null.c:7)
==43165==  If you believe this happened as a result of a stack
==43165==  overflow in your program's main thread (unlikely but
==43165==  possible), you can try to increase the size of the
==43165==  main thread stack using the --main-stacksize= flag.
==43165==  The main thread stack size used in this run was 8388608.
==43165==
==43165== HEAP SUMMARY:
==43165==     in use at exit: 0 bytes in 0 blocks
==43165==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==43165==
==43165== All heap blocks were freed -- no leaks are possible
==43165==
==43165== For lists of detected and suppressed errors, rerun with: -s
==43165== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Violación de segmento
```

4. En GDB no se encuentra el problema de la no liberación de memoria como se puede apreciar en la imagen 4.1, ya que gdb no cuenta con funciones de asignación de memoria, sin embargo, Valgrind expone el problema de no liberar memoria después de haberla asignado como se nota en la imagen 4.2.

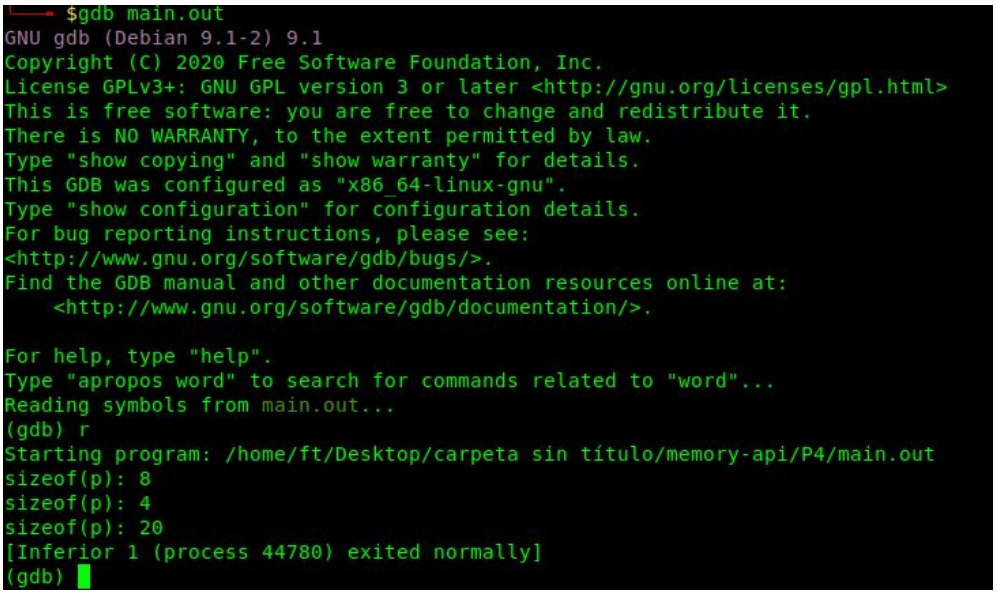

Imagen 4.1. *GDB*

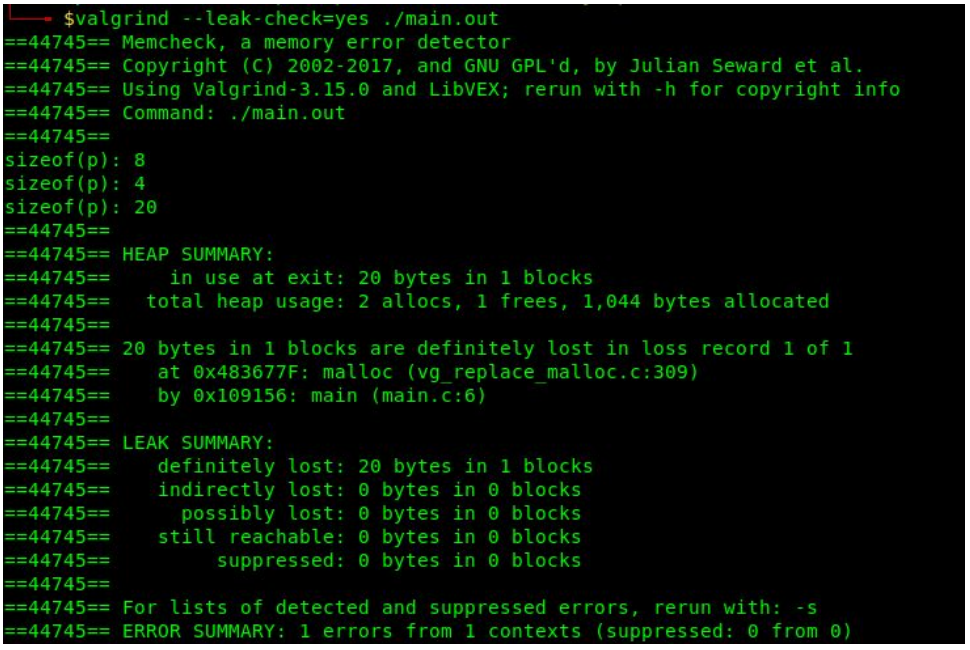

Imagen 4.2. *Valgrind*

5. El programa cuando se compila y ejecuta, realiza toda su ejecución completamente, sin embargo, al realizarle un debug con valgrind, el programa muestra que en la línea de asignación de data[100] = 0 hay un problema, ya que esta dirección no se encuentra asignada previamente por el mallloc.

```
└──╼ $valgrind --leak-check=yes ./main.out
==46542== Memcheck, a memory error detector
==46542== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==46542== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==46542== Command: ./main.out
==46542==
==46542== Invalid write of size 4
==46542==    at 0x109165: main (main.c:6)
==46542==  Address 0x4a341d0 is 0 bytes after a block of size 400 alloc'd
==46542==    at 0x483677F: malloc (vg_replace_malloc.c:309)
==46542==    by 0x109156: main (main.c:5)
==46542==
==46542==
==46542== HEAP SUMMARY:
==46542==     in use at exit: 0 bytes in 0 blocks
==46542==   total heap usage: 1 allocs, 1 frees, 400 bytes allocated
==46542==
==46542== All heap blocks were freed -- no leaks are possible
==46542==
==46542== For lists of detected and suppressed errors, rerun with: -s
==46542== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

6. Cuando se ejecuta el programa sin debugger, corre perfectamente, pero el resultado otorgado es 0, ya que no contiene datos después de ser liberado. Al usar una asignación de 30 en una posición e imprimir esta posición antes de liberar la memoria, el resultado sigue siendo 30, sin embargo, al liberarla se obtiene un resultado de 0. En valgrind se sigue obteniendo el mismo valor antes y después de liberar la memoria pero mostrando los problemas de acceder después de liberar la memoria, tal como se muestra en la siguiente imagen.

```
┌─[ft@parrot]─[~/Desktop/carpeta sin título/memory-api/P6]
└──╼ $./main.out
Este es el dato antes del free 30
Este es el dato despues del free 0
┌─[ft@parrot]─[~/Desktop/carpeta sin título/memory-api/P6]
└──╼ $valgrind --leak-check=yes ./main.out
==49667== Memcheck, a memory error detector
==49667== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==49667== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==49667== Command: ./main.out
==49667==
Este es el dato antes del free 30
==49667== Invalid read of size 4
==49667==    at 0x10919E: main (main.c:9)
==49667==  Address 0x4a34040 is 0 bytes inside a block of size 400 free'd
==49667==    at 0x48379AB: free (vg_replace_malloc.c:540)
==49667==    by 0x109199: main (main.c:8)
==49667==  Block was alloc'd at
==49667==    at 0x483677F: malloc (vg_replace_malloc.c:309)
==49667==    by 0x109166: main (main.c:5)
==49667==
Este es el dato despues del free 30
==49667==
==49667== HEAP SUMMARY:
==49667==     in use at exit: 0 bytes in 0 blocks
==49667==   total heap usage: 2 allocs, 2 frees, 1,424 bytes allocated
==49667==
==49667== All heap blocks were freed -- no leaks are possible
==49667==
==49667== For lists of detected and suppressed errors, rerun with: -s
==49667== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

7. El programa al ejecutarse lo primero que hace es liberar la memoria reservada inicialmente, cuando se hace la segunda liberación de memoria, como ya la memoria a la que apuntaba estaba liberada, entonces no tiene nada más por liberar, lo cual hace que genere un error. Al usar valgrind (el cual no es necesario), se evidencia donde fue el error y es más específico el por que, tal como se aprecia en la imagen.

```
┌─[ft@parrot]─[~/Desktop/carpeta sin título/memory-api/P7]
└──╼ $./main.out
free(): invalid pointer
Abortado
┌─[✗]─[ft@parrot]─[~/Desktop/carpeta sin título/memory-api/P7]
└──╼ $valgrind --leak-check=yes ./main.out
==51361== Memcheck, a memory error detector
==51361== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==51361== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==51361== Command: ./main.out
==51361==
==51361== Invalid free() / delete / delete[] / realloc()
==51361==    at 0x48379AB: free (vg_replace_malloc.c:540)
==51361==    by 0x10918C: main (main.c:9)
==51361==  Address 0x4a3404c is 12 bytes inside a block of size 400 free'd
==51361==    at 0x48379AB: free (vg_replace_malloc.c:540)
==51361==    by 0x109180: main (main.c:8)
==51361==  Block was alloc'd at
==51361==    at 0x483677F: malloc (vg_replace_malloc.c:309)
==51361==    by 0x109156: main (main.c:5)
==51361==
==51361==
==51361== HEAP SUMMARY:
==51361==     in use at exit: 0 bytes in 0 blocks
==51361==   total heap usage: 1 allocs, 2 frees, 400 bytes allocated
==51361==
==51361== All heap blocks were freed -- no leaks are possible
==51361==
==51361== For lists of detected and suppressed errors, rerun with: -s
==51361== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

8. Aunque utilizando realloc se pudo realizar una implementación sin problemas, se tienen que tener muchas variables que ayuden a controlar la información que entra, para así añadir un nuevo espacio de memoria. Esto tiene mucha relación con las listas ligadas, ya que no tiene un espacio finito de memoria reservada, más bien es dinámico y lo que se va necesitando se va asignando.

```
┌─[ft@parrot]─[~/Desktop/carpeta sin título/memory-api/P8]
└──╼ $valgrind --leak-check=yes ./main.out
==55919== Memcheck, a memory error detector
==55919== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==55919== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==55919== Command: ./main.out
==55919==
Vector de 1 posiciones
Ingrese un dato para la posición 0 ó 0 para terminar:
1
Se le aumenta una posición al vector
 Vector de 2 posiciones
Ingrese un dato para la posición 1 ó 0 para terminar:
2
Se le aumenta una posición al vector
 Vector de 3 posiciones
Ingrese un dato para la posición 2 ó 0 para terminar:
3
Se le aumenta una posición al vector
 Vector de 4 posiciones
Ingrese un dato para la posición 3 ó 0 para terminar:
6
Se le aumenta una posición al vector
 Vector de 5 posiciones
Ingrese un dato para la posición 4 ó 0 para terminar:
0

Datos finales del vector:

1 2 3 6 0
==55919==
==55919== HEAP SUMMARY:
==55919==     in use at exit: 0 bytes in 0 blocks
==55919==   total heap usage: 7 allocs, 7 frees, 2,108 bytes allocated
==55919==
==55919== All heap blocks were freed -- no leaks are possible
==55919==
==55919== For lists of detected and suppressed errors, rerun with: -s
==55919== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```