

Sistemas operativos y laboratorio - Práctica #2



Grupo

Diego Alexander Chavarría

Daniel Arango Obando

Alejandro Mercado Espinosa

Profesor

Jheisson Argiro Lopez

**11 de Febrero de 2022
Medellín - Antioquia**

1.

El programa genera error, ya que el puntero le está apuntando a Null y a lo que hay en esa posición de memoria intenta hacer la asignación del 0.

Cuando un fragmento de código intenta realizar una operación de lectura y escritura en una ubicación de solo lectura en la memoria o en un bloque de memoria liberado

```
→ Practica_2 git:(main) ✘ ./null
[1] 10291170 segmentation fault (core dumped) ./null
```

2.

```
Breakpoint 1, main (argc=1, argv=0x7fffffffda38) at null.c:7
7         int* p = NULL;
(gdb) print p
$1 = (int *) 0x0
(gdb) continue
Continuing.
```

```
Breakpoint 2, main (argc=1, argv=0x7fffffffda38) at null.c:8
8         *p = 0;
(gdb) print p
$2 = (int *) 0x0
(gdb) continue
Continuing.
```

```
Program received signal SIGSEGV, Segmentation fault.
0x000055555555168 in main (argc=1, argv=0x7fffffffda38) at null.c:8
8         *p = 0;
(gdb)
```

En la depuración GDB cuando establecemos el punto de parada en la línea 8 da un error similar al de la compilación sin utilizar la GDB, en este caso indica que está intentando acceder a memoria no válida.

3.

```
→ Practica_2 git:(main) ✘ valgrind --leak-check=yes ./null
==11956== Memcheck, a memory error detector
==11956== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11956== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==11956== Command: ./null
==11956==
==11956== Invalid write of size 4
==11956==   at 0x109168: main (null.c:8)
==11956==   Address 0x0 is not stack'd, malloc'd or (recently) free'd
==11956==
==11956==
==11956== Process terminating with default action of signal 11 (SIGSEGV)
==11956== Access not within mapped region at address 0x0
==11956==   at 0x109168: main (null.c:8)
==11956== If you believe this happened as a result of a stack
==11956== overflow in your program's main thread (unlikely but
==11956== possible), you can try to increase the size of the
==11956== main thread stack using the --main-stacksize= flag.
==11956== The main thread stack size used in this run was 8388608.
==11956==
==11956== HEAP SUMMARY:
==11956==   in use at exit: 0 bytes in 0 blocks
==11956==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==11956==
==11956== All heap blocks were freed -- no leaks are possible
==11956==
==11956== For lists of detected and suppressed errors, rerun with: -s
==11956== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[1]    11956 segmentation fault (core dumped) valgrind --leak-check=yes ./null
```

```
→ Practica_2 git:(main) ✘ valgrind --leak-check=yes ./null
==11956== Memcheck, a memory error detector
==11956== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11956== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==11956== Command: ./null
==11956==
==11956== Invalid write of size 4
==11956==   at 0x109168: main (null.c:8)
==11956==   Address 0x0 is not stack'd, malloc'd or (recently) free'd
==11956==
==11956==
==11956== Process terminating with default action of signal 11 (SIGSEGV)
==11956== Access not within mapped region at address 0x0
==11956==   at 0x109168: main (null.c:8)
==11956== If you believe this happened as a result of a stack
==11956== overflow in your program's main thread (unlikely but
==11956== possible), you can try to increase the size of the
==11956== main thread stack using the --main-stacksize= flag.
==11956== The main thread stack size used in this run was 8388608.
==11956==
==11956== HEAP SUMMARY:
==11956==   in use at exit: 0 bytes in 0 blocks
==11956==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==11956==
==11956== All heap blocks were freed -- no leaks are possible
==11956==
==11956== For lists of detected and suppressed errors, rerun with: -s
==11956== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[1]    11956 segmentation fault (core dumped) valgrind --leak-check=yes ./null
```

Mensaje similar al obtenido con GDB, indicando que se intenta acceder a memoria no válida, además Valgrind sugiere una posible solución que es aumentar el tamaño de la pila principal

4.

```
→ Practica_2 git:(main) ✘ ./fourth
Cuantos elementos tendra el vector:3
Ingrese elemento:1
Ingrese elemento:2
Ingrese elemento:3
→ Practica_2 git:(main) ✘
```

Cuando el programa se ejecuta, el compila con normalidad y no arroja errores.

Cuando se usa GDB:

```
For help, type "help".
Type "apropos word" to search for commands related to "w
Reading symbols from fourth...
(gdb) break 18
Breakpoint 1 at 0x1255: file fourth.c, line 18.
(gdb) run
Starting program: /home/dchavarria/Documents/Universidad
Cuantos elementos tendra el vector:3
Ingrese elemento:1
Ingrese elemento:2
Ingrese elemento:3

Breakpoint 1, main () at fourth.c:18
18    }
(gdb) ■
```

GDB tampoco detecta problemas como este, ya que el programa llega hasta el final sin reportar un error

```
==3117== Memcheck, a memory error detector
==3117== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3117== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3117== Command: ./a.out
==3117==
Cuantos elementos tendra el vector:5
Ingrese elemento:1
Ingrese elemento:1
Ingrese elemento:1
Ingrese elemento:1
Ingrese elemento:1
==3117==
==3117== HEAP SUMMARY:
==3117==     in use at exit: 20 bytes in 1 blocks
==3117==   total heap usage: 3 allocs, 2 frees, 2,068 bytes allocated
==3117==
==3117== 20 bytes in 1 blocks are definitely lost in loss record 1 of 1
==3117==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==3117==    by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==3117==
==3117== LEAK SUMMARY:
```

```
==3117==  
==3117== LEAK SUMMARY:  
==3117==   definitely lost: 20 bytes in 1 blocks  
==3117==   indirectly lost: 0 bytes in 0 blocks  
==3117==   possibly lost: 0 bytes in 0 blocks  
==3117==   still reachable: 0 bytes in 0 blocks  
==3117==       suppressed: 0 bytes in 0 blocks  
==3117==  
==3117== For lists of detected and suppressed errors, rerun with: -s  
==3117== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)  
daniel@daniel-VirtualBox:~/Descargas/Practica 2$
```

Hay una filtraciones en la memoria, indicando que hay 20 bytes que se están perdiendo al no liberar la memoria

5.

Cuando se ejecuta, no muestra ningún error, se compila con normalidad:

```
→ Practica_2 git:(main) ✘ gcc -o five five.c  
→ Practica_2 git:(main) ✘
```

GDB:

```
Breakpoint 5, main () at five.c:10  
10      }  
(gdb) c  
Continuing.  
[Inferior 1 (process 15623) exited normally]  
(gdb)
```

GDB se ejecuta con normalidad, al finalizar el programa, la posición de memoria a la que apuntaba el array y el valor de data[100] se quedan cargados con esos valores porque no se liberó, esto se puede apreciar en la siguiente imagen

```
Breakpoint 5, main () at five.c:10  
10      }  
(gdb) p data  
$6 = (int *) 0x55555555592a0  
(gdb) p data[100]  
$7 = 0  
(gdb)
```

Valgrind:

```
==3445== Memcheck, a memory error detector
==3445== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3445== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3445== Command: ./a.out
==3445==
==3445== Invalid write of size 4
==3445==   at 0x109174: main (in /home/daniel/Descargas/Practica 2/a.out)
==3445==   Address 0x4a501d0 is 0 bytes after a block of size 400 alloc'd
==3445==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==3445==   by 0x109165: main (in /home/daniel/Descargas/Practica 2/a.out)
==3445==
==3445== HEAP SUMMARY:
==3445==   in use at exit: 400 bytes in 1 blocks
==3445==   total heap usage: 1 allocs, 0 frees, 400 bytes allocated
==3445==
==3445== 400 bytes in 1 blocks are definitely lost in loss record 1 of 1
==3445==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==3445==   by 0x109165: main (in /home/daniel/Descargas/Practica 2/a.out)
==3445==
==3445== LEAK SUMMARY:
==3445==   definitely lost: 400 bytes in 1 blocks
==3445==   indirectly lost: 0 bytes in 0 blocks
==3445==   possibly lost: 0 bytes in 0 blocks
==3445==   still reachable: 0 bytes in 0 blocks
==3445==   suppressed: 0 bytes in 0 blocks
==3445==
```

6.

Codifique un programa que asigne un array de enteros (como arriba), luego lo libere, y entonces intente imprimir el valor de un elemento del array. ¿El programa corre?, ¿Qué pasa cuando hace uso de valgrind?

El programa corre, e imprime el valor de data[4] a pesar de haber sido liberado el array data con la instrucción free(), cuando se usa GDB se aprecia que efectivamente luego de haber liberado el array, la cuarta posición de este queda cargada con el valor que se le había asignado, en este caso es el 10

```
Breakpoint 3, main () at six.c:10
10      printf("\n %d", (data[4]));
(gdb) p [data]
A syntax error in expression, near `[data]'.
(gdb) p data
$7 = (int *) 0x5555555592a0
(gdb) p data[4]
$8 = 10
(gdb) c
Continuing.
```

```
daniel@daniel-VirtualBox:~/Descargas/Practica 2$ valgrind --leak-check=yes ./a.out
==3555== Memcheck, a memory error detector
==3555== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3555== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3555== Command: ./a.out
==3555==
==3555== Invalid read of size 4
==3555==     at 0x1091CC: main (in /home/daniel/Descargas/Practica 2/a.out)
==3555==   Address 0x4a50050 is 16 bytes inside a block of size 20 free'd
==3555==     at 0x483CA3F: free (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==3555==     by 0x1091C3: main (in /home/daniel/Descargas/Practica 2/a.out)
==3555==   Block was alloc'd at
==3555==     at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==3555==     by 0x1091A5: main (in /home/daniel/Descargas/Practica 2/a.out)
==3555==

10==3555==
==3555== HEAP SUMMARY:
==3555==     in use at exit: 0 bytes in 0 blocks
==3555==   total heap usage: 2 allocs, 2 frees, 1,044 bytes allocated
==3555== All heap blocks were freed -- no leaks are possible
==3555==
==3555== For lists of detected and suppressed errors, rerun with: -s
==3555== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Una lectura no válida significa que la ubicación de memoria que el proceso intentaba leer está fuera de las direcciones de memoria que están disponibles para el proceso . tamaño 4 significa que el proceso estaba tratando de leer 4 bytes.

8.

Código sin errores valgrind:

```
printf("Desea agregar otro elemento? S/N:");
scanf("%i", &next);
if(next == 1){
    index++;
    tam += 4;
    numbers = (int*) realloc(numbers, tam);
}

le (next != 0);
//free(&contenido del vector dinamico);
```

```

daniel@daniel-VirtualBox:~/Descargas/Practica 2$ valgrind --leak-check=yes ./Punto8
==7805== Memcheck, a memory error detector
==7805== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7805== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==7805== Command: ./Punto8
==7805==

Ingrese elemento:5
Desea agregar otro elemento? S/N:1
Ingrese elemento:4
Desea agregar otro elemento? S/N:1
Ingrese elemento:2
Desea agregar otro elemento? S/N:1
Ingrese elemento:4
Desea agregar otro elemento? S/N:1
Ingrese elemento:2
Desea agregar otro elemento? S/N:0
Contenido del vector dinamico:5 4 2 4 2 ==7805==
==7805== HEAP SUMMARY:
==7805==     in use at exit: 0 bytes in 0 blocks
==7805==   total heap usage: 7 allocs, 7 frees, 2,108 bytes allocated
==7805== All heap blocks were freed -- no leaks are possible
==7805== For lists of detected and suppressed errors, rerun with: -s
==7805== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
daniel@daniel-VirtualBox:~/Descargas/Practica 2$ █

```

El vector como vemos funciona sin problemas alojando cada uno de los números que le introducimos por consola, a la hora de ejecutar **valgrind** nos damos cuenta que no tenemos ningún error de memoria, debido a que cada que queremos introducir un nuevo número, con la función **realloc** estamos reservando la cantidad necesaria de bits para el nuevo elemento. Ahora revisemos qué pasaría si no usáramos la función **Realloc**:

Comentando **Realloc**:

```

//next != 0) {
    index++;
    tam += 4;
    //numbers = (int*) realloc(numbers, tam);

    (next != 0);
    f("Contenido del vector dinamico:");
}

```

```

Desea agregar otro elemento? S/N:1
Ingrese elemento:2
==7855== Invalid write of size 4
==7855==   at 0x48C64B5: __vfscanf_internal (vfscanf-internal.c:1895)
==7855==   by 0x48C12E1: __isoc99_scanf (isoc99_scanf.c:30)
==7855==   by 0x109242: main (in /home/daniel/Descargas/Practica 2/Punto8)
==7855== Address 0x4a50044 is 0 bytes after a block of size 4 alloc'd
==7855==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==7855==   by 0x109205: main (in /home/daniel/Descargas/Practica 2/Punto8)
==7855==

Desea agregar otro elemento? S/N:1
Ingrese elemento:5
Desea agregar otro elemento? S/N:1
Ingrese elemento:3
Desea agregar otro elemento? S/N:1
Ingrese elemento:8
Desea agregar otro elemento? S/N:0
==7855== Invalid read of size 4
==7855==   at 0x1092B1: main (in /home/daniel/Descargas/Practica 2/Punto8)
==7855== Address 0x4a50044 is 0 bytes after a block of size 4 alloc'd
==7855==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==7855==   by 0x109205: main (in /home/daniel/Descargas/Practica 2/Punto8)
==7855==

Contenido del vector dinamico:5 2 5 3 8 ==7855==
==7855== HEAP SUMMARY:
==7855==     in use at exit: 0 bytes in 0 blocks
==7855==   total heap usage: 3 allocs, 3 frees, 2.052 bytes allocated

```

Como vemos ahora tenemos problemas con valgrind, esto es debido a que no estamos haciendo el realloc y por ende tenemos errores en lectura y escritura del vector al no contemplar los bits necesarios para el vector.

Se puede comparar con una lista ligada debido a que podemos generar un vector con los espacios que el usuario solicite, sin necesidad de definir un límite inicial.

9.

El error ocurre debido a que se está retornando la dirección de memoria de una variable que tiene un ámbito local dentro del bloque del método, por lo tanto una vez que el método haga return no va a existir más este almacenamiento. Esto lo podemos apreciar con valgrind debido a que nos muestra el resultado de string una vez se termine su valor local en el método:

```

→ Practica_2 git:(main) ✘ gcc -o nine -g nine.c
nine.c: In function 'getString':
nine.c:6:9: warning: function returns address of local variable [-Wreturn-local-addr]
    6 |   return message;
          ^~~~~~
→ Practica_2 git:(main) ✘

```

```
daniel@daniel-VirtualBox:~/Descargas/Practica 2$ valgrind --leak-check=yes ./a.out
==4130== Memcheck, a memory error detector
==4130== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4130== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4130== Command: ./a.out
==4130==
String: (null)
==4130==
==4130== HEAP SUMMARY:
==4130==     in use at exit: 0 bytes in 0 blocks
==4130==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4130==
==4130== All heap blocks were freed -- no leaks are possible
==4130==
==4130== For lists of detected and suppressed errors, rerun with: -s
==4130== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
daniel@daniel-VirtualBox:~/Descargas/Practica 2$ █
```

10.

Valgrind:

```
==4237== Memcheck, a memory error detector
==4237== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4237== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4237== Command: ./a.out
==4237==
==4237== Invalid write of size 4
==4237==   at 0x1091B7: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==   Address 0x4a50040 is 0 bytes inside a block of size 3 alloc'd
==4237==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==4237==   by 0x10918F: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== Conditional jump or move depends on uninitialised value(s)
==4237==   at 0x48D5AD8: __vfprintf_internal (vfprintf-internal.c:1687)
==4237==   by 0x48BFEBE: printf (printf.c:33)
==4237==   by 0x1091E4: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== Use of uninitialised value of size 8
==4237==   at 0x48B981B: _itoa_word (_itoa.c:179)
==4237==   by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==4237==   by 0x48BFEBE: printf (printf.c:33)
==4237==   by 0x1091E4: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== Conditional jump or move depends on uninitialised value(s)
==4237==   at 0x48B982D: _itoa_word (_itoa.c:179)
==4237==   by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==4237==   by 0x48BFEBE: printf (printf.c:33)
==4237==   by 0x1091E4: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== Conditional jump or move depends on uninitialised value(s)
==4237==   at 0x48D63A8: __vfprintf_internal (vfprintf-internal.c:1687)
==4237==   by 0x48BFEBE: printf (printf.c:33)
==4237==   by 0x1091E4: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== Conditional jump or move depends on uninitialised value(s)
==4237==   at 0x48D586E: __vfprintf_internal (vfprintf-internal.c:1687)
==4237==   by 0x48BFEBE: printf (printf.c:33)
==4237==   by 0x1091E4: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
0: 0
==4237== Invalid read of size 4
==4237==   at 0x1091CD: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==   Address 0x4a50044 is 1 bytes after a block of size 3 alloc'd
==4237==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==4237==   by 0x10918F: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
1: 1
2: 4
```

```
1: 1
2: 4
==4237==
==4237== HEAP SUMMARY:
==4237==     in use at exit: 3 bytes in 1 blocks
==4237== total heap usage: 2 allocs, 1 frees, 1,027 bytes allocated
==4237==
==4237== 3 bytes in 1 blocks are definitely lost in loss record 1 of 1
==4237==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==4237==    by 0x10918F: main (in /home/daniel/Descargas/Practica 2/a.out)
==4237==
==4237== LEAK SUMMARY:
==4237==    definitely lost: 3 bytes in 1 blocks
==4237==    indirectly lost: 0 bytes in 0 blocks
==4237==    possibly lost: 0 bytes in 0 blocks
==4237==    still reachable: 0 bytes in 0 blocks
==4237==    suppressed: 0 bytes in 0 blocks
==4237==
==4237== Use --track-origins=yes to see where uninitialized values come from
==4237== For lists of detected and suppressed errors, rerun with: -s
==4237== ERROR SUMMARY: 11 errors from 8 contexts (suppressed: 0 from 0)
daniel@daniel-VirtualBox:~/Descargas/Practica 2$
```

Valgrind: Inicialmente se detecta la pérdida de 3 posiciones de memoria, dado que se está separando y no se está usando (0 bytes alojados en ese espacio). El error invalid write of size 4, se refiere a que el programa intenta escribir en una posición de memoria más allá de la asignada inicialmente, esto debido a un error al asignar posición de memoria al malloc. Esto puede ser tomado como sugerencia para hacer un refactor de ese malloc y crear los espacios de memoria para cada unidad del array del tamaño del tipo de dato que se va a alojar. Con GDB esto no se detecta, ya que al imprimir cada posición de memoria, se obtiene el valor, y no hay un error explícito que se muestre.

11.

```
==4721== Memcheck, a memory error detector
==4721== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4721== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4721== Command: ./a.out
==4721==
==4721== Invalid write of size 4
==4721==   at 0x1091D7: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==   Address 0x4a50040 is 0 bytes inside a block of size 3 alloc'd
==4721==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==4721==   by 0x1091AF: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==
==4721== Conditional jump or move depends on uninitialised value(s)
==4721==   at 0x48D5AD8: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==
==4721== Use of uninitialised value of size 8
==4721==   at 0x48B981B: _itoa_word (_itoa.c:179)
==4721==   by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==

==4721== Conditional jump or move depends on uninitialised value(s)
==4721==   at 0x48D5AD8: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==
==4721== Use of uninitialised value of size 8
==4721==   at 0x48B981B: _itoa_word (_itoa.c:179)
==4721==   by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==

==4721== Conditional jump or move depends on uninitialised value(s)
==4721==   at 0x48B982D: _itoa_word (_itoa.c:179)
==4721==   by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==

==4721== Conditional jump or move depends on uninitialised value(s)
==4721==   at 0x48D63A8: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==

==4721== Conditional jump or move depends on uninitialised value(s)
==4721==   at 0x48D586E: __vfprintf_internal (vfprintf-internal.c:1687)
==4721==   by 0x48BFEBE: printf (printf.c:33)
==4721==   by 0x109204: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==

0: 0
```

```
?1== Invalid read of size 4
?1==   at 0x1091ED: main (in /home/daniel/Descargas/Practica 2/a.out)
?1==   Address 0x4a50044 is 1 bytes after a block of size 3 free'd
?1==   at 0x483CA3F: free (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload
?1==     check-amd64-linux.so)
?1==   by 0x109210: main (in /home/daniel/Descargas/Practica 2/a.out)
?1== Block was alloc'd at
?1==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
?1==     emcheck-amd64-linux.so)
?1==   by 0x1091AF: main (in /home/daniel/Descargas/Practica 2/a.out)
?1==
```

El problema en el código es que está reservando 3 posiciones de memoria para el vector heights, y en cada una de ellas se esta almacenando un entero que ocupa 4 bytes, en GDB no se rastreó facil este error, pero en Valgrind se ofrece una buena descripción del mismo. Si se hace un malloc preparado para almacenar 3 enteros completos no se presenta el error.

12.

Código con error:

```
int main(int argc, char* argv[]){
    const int NUM_HEIGHTS = 10;
    int *heights = malloc(NUM_HEIGHTS*sizeof(*heights));
    for(int i=0; i<NUM_HEIGHTS; i++){
        if(heights == NULL){
            heights = malloc((NUM_HEIGHTS*sizeof(*heights)));
        }
        free(heights);
    }
```

Corregido:

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main(int argc, char* argv[]){
5     const int NUM_HEIGHTS = 10;
6     int *heights = malloc(NUM_HEIGHTS*sizeof(*heights));
7     for(int i=0; i<NUM_HEIGHTS; i++){
8         if(heights == NULL){
9             heights = malloc(NUM_HEIGHTS*sizeof(*heights));
10        }
11    }
12    free(heights);
13 }
```

```
--5900== Memcheck, a memory error detector
--5900== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
--5900== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
--5900== Command: ./a.out
--5900==
--5900==
--5900== HEAP SUMMARY:
--5900==       in use at exit: 40 bytes in 1 blocks
--5900==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
--5900==
--5900== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
--5900==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
--5900==    by 0x109193: main (in /home/daniel/Descargas/Practica 2/a.out)
--5900==
--5900== LEAK SUMMARY:
--5900==   definitely lost: 40 bytes in 1 blocks
--5900==   indirectly lost: 0 bytes in 0 blocks
--5900==   possibly lost: 0 bytes in 0 blocks
--5900==   still reachable: 0 bytes in 0 blocks
--5900==           suppressed: 0 bytes in 0 blocks
--5900==
--5900== For lists of detected and suppressed errors, rerun with: -s
--5900== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

```

==4721== Invalid free() / delete / delete[] / realloc()
==4721==    at 0x483CA3F: free (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==4721==    by 0x109210: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==    Address 0x4a50040 is 0 bytes inside a block of size 3 free'd
==4721==    at 0x483CA3F: free (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==4721==    by 0x109210: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==  Block was alloc'd at
==4721==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==4721==    by 0x1091AF: main (in /home/daniel/Descargas/Practica 2/a.out)
==4721==
2: 4
==4721==
==4721== HEAP SUMMARY:
==4721==     in use at exit: 0 bytes in 0 blocks
==4721==   total heap usage: 2 allocs, 4 frees, 1,027 bytes allocated
==4721==
==4721== All heap blocks were freed -- no leaks are possible
==4721==
==4721== Use --track-origins=yes to see where uninitialized values come from
==4721== For lists of detected and suppressed errors, rerun with: -s
==4721== ERROR SUMMARY: 12 errors from 8 contexts (suppressed: 0 from 0)

```

El error es muy similar al del punto 11, la diferencia es que aparece uno adicional “Invalid Free” que indica la memoria se está liberando de forma incorrecta, en este caso específico se está intentando liberar la memoria más de una vez. Si bien valgrind da muchas indicaciones acerca del error, fue más sencillo encontrarlo durante la compilación del programa que directamente arrojaba error de compilación

```

→ Practica_2 git:(main) ✘ ./eleven
0: 0
1: 1
free(): double free detected in tcache 2
[1]    15612 abort (core dumped) ./eleven

```

13.

En el resumen de la ejecución se obtienen las siguientes estadísticas:

```

Tamaño de la memoria en la región STACK: 132
Número de cambios de contexto realizados (voluntarios - no voluntarios): 23 - 0
==27270==
==27270== HEAP SUMMARY:
==27270==     in use at exit: 944 bytes in 2 blocks
==27270==   total heap usage: 7 allocs, 5 frees, 5,864 bytes allocated
==27270==
==27270== LEAK SUMMARY:
==27270==     definitely lost: 0 bytes in 0 blocks
==27270==     indirectly lost: 0 bytes in 0 blocks
==27270==     possibly lost: 0 bytes in 0 blocks
==27270==     still reachable: 944 bytes in 2 blocks
==27270==           suppressed: 0 bytes in 0 blocks
==27270== Rerun with --leak-check=full to see details of leaked memory
==27270==
==27270== Use --track-origins=yes to see where uninitialized values come from
==27270== For lists of detected and suppressed errors, rerun with: -s
==27270== ERROR SUMMARY: 29 errors from 18 contexts (suppressed: 0 from 0)

```

Donde se resaltan 944 bytes en 2 bloques marcados como “still reachable”. Corresponden mayoritariamente a errores de tipo Conditional jump or move depends on uninitialised values. Al trackear los errores, se encuentra que en unas instrucciones de tipo strlen se están usando variables que nunca fueron inicializadas, tal como se muestra en la siguiente imagen

```
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0A2: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0B7: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0A2: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0A2: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0B7: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0A2: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== Conditional jump or move depends on uninitialised value(s)
--27270== at 0x483F0B7: strcpy [in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so]
--27270== by 0x109AB0: createPinfo [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270== by 0x1094BA: main [in /home/dchavarria/Documents/Universidad/2021-2/Sistemas operativos/LabFinalSO/psinfo-punto2 (3)/psinfo-punto2/psinfo]
--27270==
```

El error se corrige fácilmente asignando valores iniciales a las variables:

```
char path[40], line[100], *p, stateChar[100], Name[100], VmExe[100], VmData[100], VmStk[100], VmSize[100];
FILE *statust;
char buf[100];
int voluntary, nonvoluntary;
snprintf(path, 40, "/proc/%d/status", pid);
statusf = fopen(path, "r");
if (statusf == NULL)
    return -1;

while (fgets(buf, sizeof buf, statusf) != NULL)
{
    sscanf(buf, "State: %s", stateChar);
    sscanf(buf, "Name: %s", Name);
    sscanf(buf, "VmSize: %s", VmSize);
    sscanf(buf, "VmExe: %s", VmExe);
    sscanf(buf, "VmData: %s", VmData);
    sscanf(buf, "VmStk: %s", VmStk);
    sscanf(buf, "voluntary_ctxt_switches: %d", &voluntary);
    sscanf(buf, "nonvoluntary_ctxt_switches: %d", &nonvoluntary);
}
```

14.

Massif

Es un perfilador de montículos Mide cuánta memoria de almacenamiento dinámico utiliza su programa. Esto incluye tanto el espacio útil como los bytes adicionales asignados para fines de contabilidad y alineación. También puede medir el tamaño de las pilas de su programa.

Beneficios:

Puede acelerar su programa: un programa pequeño puede interactuar mejor con los cachés de la máquina y evitará la paginación.

Si su programa usa mucha memoria, reducirá la posibilidad de que agote el espacio de intercambio de su máquina.

Además, hay ciertas fugas de espacio que no son detectadas por los detectores de fugas tradicionales, como Memcheck. Esto se debe a que la memoria en realidad nunca se pierde, queda un puntero en ella, pero no está en uso. Los programas que tienen fugas como esta pueden aumentar innecesariamente la cantidad de memoria que utilizan con el tiempo. Massif puede ayudar a identificar estas fugas.

Es importante destacar que Massif no solo le dice cuánta memoria de almacenamiento dinámico está utilizando su programa, sino que también brinda información muy detallada que indica qué partes de su programa son responsables de asignar la memoria de almacenamiento dinámico. [1]

Comando para ejecutar: `valgrind --tool=massif prog`

msprint:

En primer lugar, en cuanto a las otras herramientas de Valgrind, debe compilar con información (opción -g). No debería importar mucho el nivel de optimización con el que compile su programa, ya que es poco probable que esto afecte el uso de la memoria del heap.

Luego, debe ejecutar Massif para recopilar la información y luego ejecutar `ms_print` para presentar el resultado de forma legible.[1]

Esta opción produce una gráfica que muestra el consumo de memoria durante la ejecución del programa, adicionalmente produce información detallada sobre los puntos responsables de la asignación en varios puntos del programa, incluso en el punto de asignación máxima de memoria.

Opciones de uso más comunes

--heap

Especifica si se debe realizar la creación de perfiles de almacenamiento dinámico. El valor predeterminado es yes. La creación de perfiles de almacenamiento dinámico se puede desactivar configurando esta opción en no.

--heap-admin

Especifica la cantidad de bytes por bloque que se usará para la administración cuando la creación de perfiles de montón esté habilitada. El valor predeterminado es de 8 bytes por bloque.

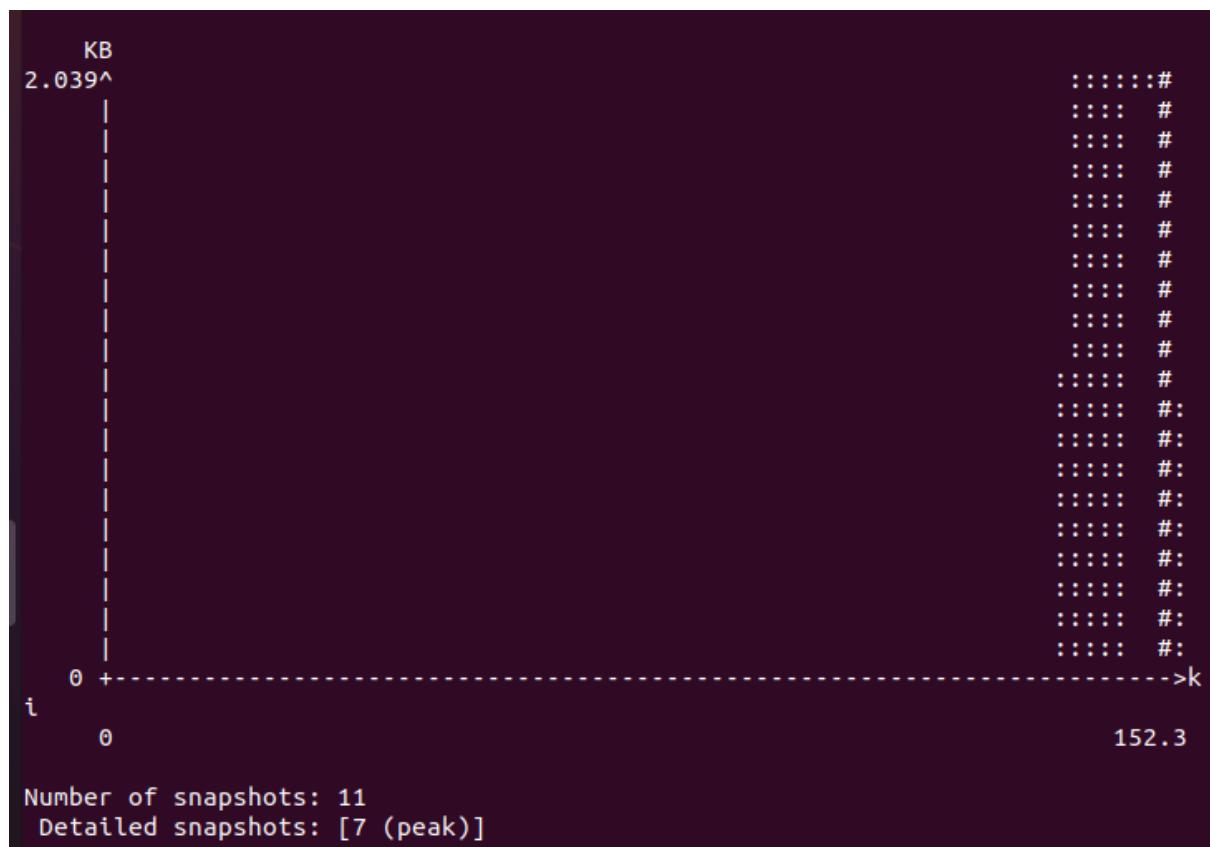
--stacks

Especifica si se debe realizar la creación de perfiles de pila. El valor predeterminado es no(deshabilitado). Para habilitar la creación de perfiles de pila, establezca esta opción en yes, pero tenga en cuenta que hacerlo ralentizará mucho a Massif. También tenga en cuenta que Massif asume que la pila principal tiene un tamaño cero al inicio para indicar mejor el tamaño de la porción de la pila sobre la que tiene control el programa que se perfila.

--time-unit

Especifica la unidad de tiempo utilizada para la generación de perfiles. Hay tres valores válidos para esta opción: instrucciones ejecutadas (i), el valor por defecto, que es útil en la mayoría de los casos; tiempo real (ms, en milisegundos), que puede ser útil en ciertos casos; y bytes asignados/desasignados en el montón y/o pila (B), lo cual es útil para programas de ejecución muy corta y para fines de prueba, porque es el más reproducible en diferentes máquinas. Esta opción es útil cuando se grafica la salida de Massif con ms_print.

Ejemplo ejecutado sobre le punto 8:



```

-----
-
 n      time(i)      total(B)   useful-heap(B) extra-heap(B)   stacks(B)
-----
-
 0          0          0          0          0          0
 1     137,934        24          4         20          0
 2     138,229       1,056        1,028        28          0
 3     139,721       2,088        2,052        36          0
 4     141,817       2,088        2,056        32          0
 5     144,484       2,088        2,060        28          0
 6     147,151       2,088        2,064        24          0
 7     152,696       2,088        2,064        24          0
98.85% (2,064B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.
->98.08% (2,048B) 0x48D6E83: _IO_file_doallocate (filedoalloc.c:101)
| ->98.08% (2,048B) 0x48E704F: _IO_dallocbuf (genops.c:347)
|   ->49.04% (1,024B) 0x48E5E23: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:486
)
|   | ->49.04% (1,024B) 0x48E7105: _IO_default_uflow (genops.c:362)
|   |   ->49.04% (1,024B) 0x48B93FF: __vfscanf_internal (vfscanf-internal.c:627
)
|   |   ->49.04% (1,024B) 0x48B82E1: __isoc99_scanf (isoc99_scanf.c:30)
|   |   ->49.04% (1,024B) 0x109262: main (in /home/daniel/Descargas/Practic
a 2/Punto8)
|   |
|   ->49.04% (1,024B) 0x109262: main (in /home/daniel/Descargas/Practic
a 2/Punto8)
|   |
|   ->49.04% (1,024B) 0x48E60AF: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)
|   ->49.04% (1,024B) 0x48E4834: _IO_new_file_xsputn (fileops.c:1244)
|   ->49.04% (1,024B) 0x48E4834: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:11
97)
|   ->49.04% (1,024B) 0x48CBAF1: __vfprintf_internal (vfprintf-internal.c
:1373)
|   ->49.04% (1,024B) 0x48B6EBE: printf (printf.c:33)
|   ->49.04% (1,024B) 0x10923A: main (in /home/daniel/Descargas/Pract
ica 2/Punto8)
|
->00.77% (16B) in 1+ places, all below ms_print's threshold (01.00%)

-----
-
 n      time(i)      total(B)   useful-heap(B) extra-heap(B)   stacks(B)
-----
-
 8     152,696        2,064        2,048        16          0
 9     155,936        1,032        1,024         8          0
10     155,970          0          0          0          0
daniel@daniel-VirtualBox:~/Descargas/Practica 2$
```