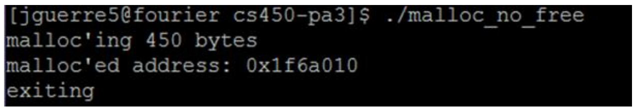
James Guerrera-Sapone - A20365203
Robert Judka - A20348051

Finding Memory Leaks with *gdb* and *Valgrind*

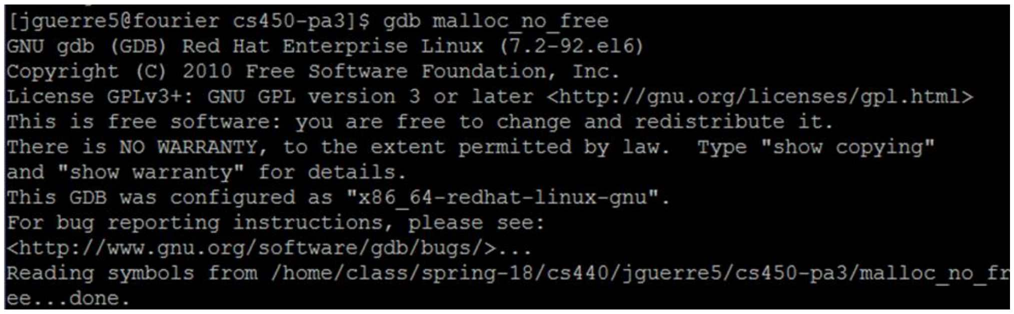
1. **malloc() but forget to free()**

   Running the program prints a message indicating the memory address of the allocated memory. Since it is never freed there is a memory leak, which means that the space is never deallocated. This can be found in *gdb* by using breakpoints and checking stack frames. Checking the local variables after the program has returned reveals that the variable's memory address is still valid. Using *Valgrind* it is immediately apparent that there is a problem with the program. It easily finds that 450 bytes were lost due to a memory leak and it can point out the allocation that caused the problem.



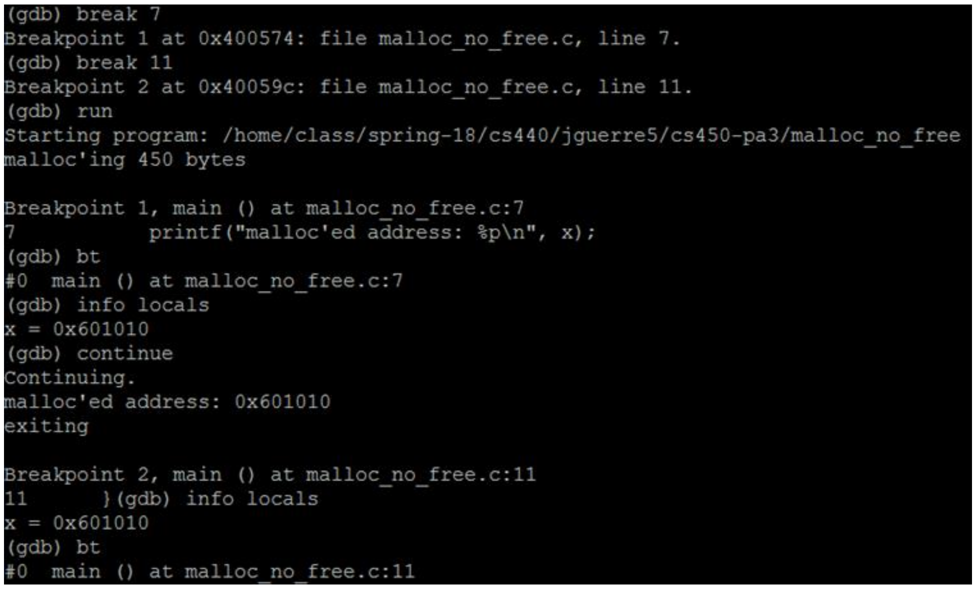
```
[jguerre5@fourier cs450-pa3]$ ./malloc_no_free
malloc'ing 450 bytes
malloc'ed address: 0x1f6a010
exiting
```

Running the program, displaying newly allocated memory address then exiting.



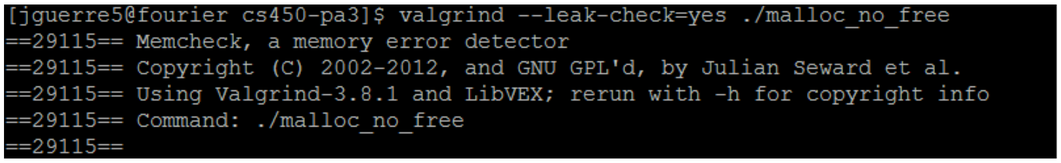
```
[jguerre5@fourier cs450-pa3]$ gdb malloc_no_free
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-92.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/class/spring-18/cs440/jguerre5/cs450-pa3/malloc_no_fr
ee...done.
```

Starting program in *gdb*.



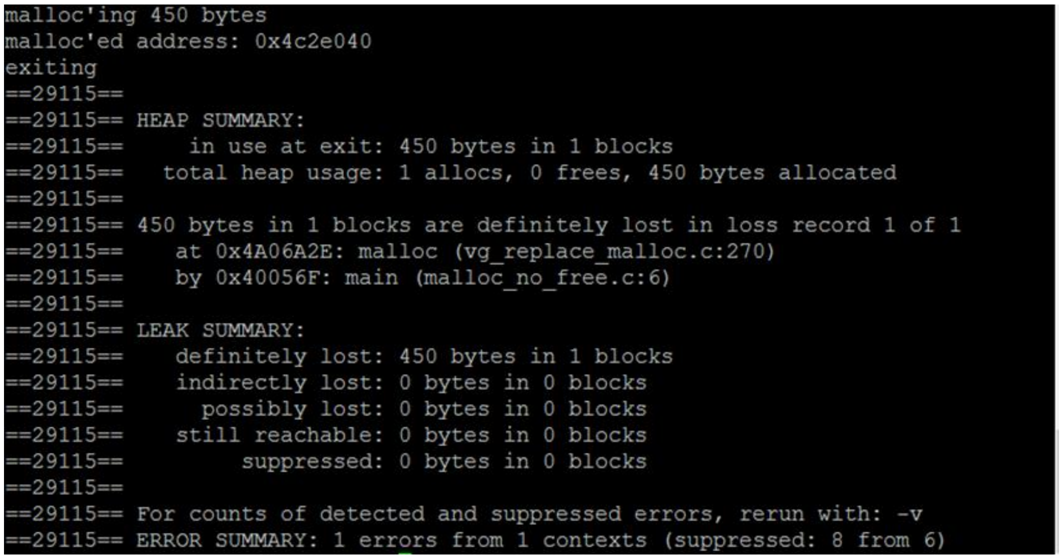
```
(gdb) break 7
Breakpoint 1 at 0x400574: file malloc_no_free.c, line 7.
(gdb) break 11
Breakpoint 2 at 0x40059c: file malloc_no_free.c, line 11.
(gdb) run
Starting program: /home/class/spring-18/cs440/jguerre5/cs450-pa3/malloc_no_free
malloc'ing 450 bytes

Breakpoint 1, main () at malloc_no_free.c:7
7           printf("malloc'ed address: %p\n", x);
(gdb) bt
#0  main () at malloc_no_free.c:7
(gdb) info locals
x = 0x601010
(gdb) continue
Continuing.
malloc'ed address: 0x601010
exiting

Breakpoint 2, main () at malloc_no_free.c:11
11      }(gdb) info locals
x = 0x601010
(gdb) bt
#0  main () at malloc_no_free.c:11
```

Setting breakpoints in *gdb* based on line number then running the program through *gdb*. The local variables are checked using "*info locals"* at each break point.



```
[jguerre5@fourier cs450-pa3]$ valgrind --leak-check=yes ./malloc_no_free
==29115== Memcheck, a memory error detector
==29115== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==29115== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==29115== Command: ./malloc_no_free
==29115==
```

Running *Valgrind* on the program to check for memory leaks.

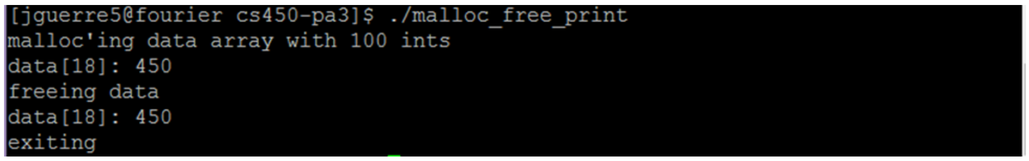


```
malloc'ing 450 bytes
malloc'ed address: 0x4c2e040
exiting
==29115==
==29115== HEAP SUMMARY:
==29115==     in use at exit: 450 bytes in 1 blocks
==29115==   total heap usage: 1 allocs, 0 frees, 450 bytes allocated
==29115==
==29115== 450 bytes in 1 blocks are definitely lost in loss record 1 of 1
==29115==    at 0x4A06A2E: malloc (vg_replace_malloc.c:270)
==29115==    by 0x40056F: main (malloc_no_free.c:6)
==29115==
==29115== LEAK SUMMARY:
==29115==    definitely lost: 450 bytes in 1 blocks
==29115==    indirectly lost: 0 bytes in 0 blocks
==29115==      possibly lost: 0 bytes in 0 blocks
==29115==    still reachable: 0 bytes in 0 blocks
==29115==         suppressed: 0 bytes in 0 blocks
==29115==
==29115== For counts of detected and suppressed errors, rerun with: -v
==29115== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 8 from 6)
```

Valgrind showing the memory leak in the program.

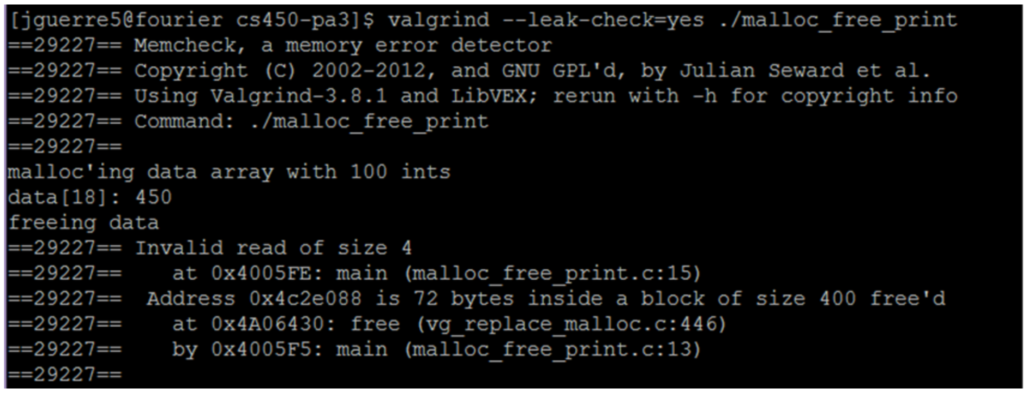
2. **malloc() array, free() array, and printf() element of array**

The program runs and returns the same value that it did before the data was freed. *Valgrind* acknowledges that the data was freed and that there are no memory leaks. However, *Valgrind* complains that there was an invalid read at the line where the data is printed after it is freed. The message states that the data being accessed is within a block of data that was freed in a prior line of code. *Valgrind* says that there is 1 error from 1 context referring to the invalid read.



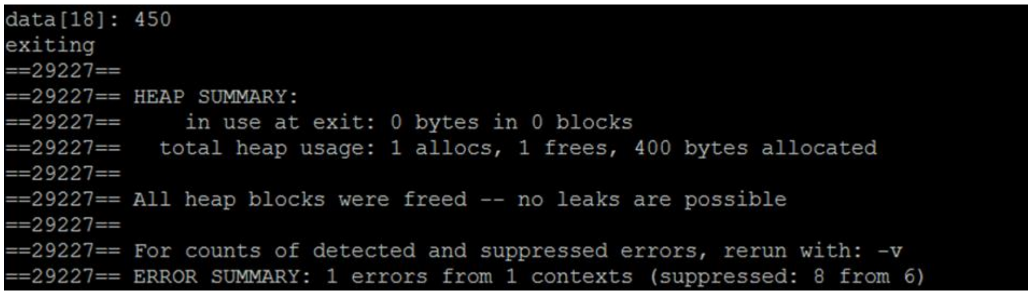
```
[jguerre5@fourier cs450-pa3]$ ./malloc_free_print
malloc'ing data array with 100 ints
data[18]: 450
freeing data
data[18]: 450
exiting
```

Running the program, displaying the value before and after freeing the array.



```
[jguerre5@fourier cs450-pa3]$ valgrind --leak-check=yes ./malloc_free_print
==29227== Memcheck, a memory error detector
==29227== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==29227== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==29227== Command: ./malloc_free_print
==29227==
malloc'ing data array with 100 ints
data[18]: 450
freeing data
==29227== Invalid read of size 4
==29227==    at 0x4005FE: main (malloc_free_print.c:15)
==29227==  Address 0x4c2e088 is 72 bytes inside a block of size 400 free'd
==29227==    at 0x4A06430: free (vg_replace_malloc.c:446)
==29227==    by 0x4005F5: main (malloc_free_print.c:13)
==29227==
```

Running *Valgrind* on the program. *Valgrind* detects an invalid read caused by freeing the array.



```
data[18]: 450
exiting
==29227==
==29227== HEAP SUMMARY:
==29227==     in use at exit: 0 bytes in 0 blocks
==29227==   total heap usage: 1 allocs, 1 frees, 400 bytes allocated
==29227==
==29227== All heap blocks were freed -- no leaks are possible
==29227==
==29227== For counts of detected and suppressed errors, rerun with: -v
==29227== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 8 from 6)
```

The rest of *Valgrind's* output where it reports one error in 1 context.