Dan Hervé

CSCD 340

Steiner

Lab 1 Problem 1

a) Does the stack grow up or down? How do you know? Justify your answer.

The stack grows downward. This can be observed by noting the difference between the variable x and its ptr, where ptr, initialized second, has a lower address. Also val, initialized after x, has a lower address.

b) What version of GCC are you using?

GCC version 4.6.4

c) What version of Linux are you using?

Linux cslinux 0.13.0-36-generic

d) What is odd about how memory is arranged compared to the declarations?

Generally each declaration is separated by the separation between declarations is not constant. Sometimes the declarations are separated by 4 bytes and sometimes by 8 bytes

e) Run the program twice and each time construct a memory map.

0x9dc010	word
0x7fff9fa90aec	x
0x7fff9fa90ae0	val
0x7fff9fa90ad8	dptr
0x7fff9fa90ad4	array[5]
0x7fff9fa90ac0	array
0x7fff9fa90ab8	val2
0x7fff9fa90ab4	y
0x7fff9fa90aa8	dptr2
0x7fff9fa90aa0	ptr2
0x7fff9fa90a98	&word
0x7fff9fa90a90	ptr

0x21fb010	word
0x7fff892c40fc	x
0x7fff892c40f0	val
0x7fff892c40e8	dptr
0x7fff892c40e4	array[5]
0x7fff892c40d0	array
0x7fff892c40c8	val2
0x7fff892c40c4	y
0x7fff892c40b8	dptr2
0x7fff892c40b0	ptr2
0x7fff892c40a8	&word
0x7fff892c40a0	ptr

f) Did the addresses change between runs? Why or why not? Justify your answer.

They did change. This is because Linux implements ASLR(Address Space Layout Randomization) to protect from security threats.

g) How many bytes are allocated by the calloc?

10 bytes

h) How many bytes are leaked? Provide the valgrind output below.

10 bytes are leaked.

```
dherve@cslinux:~/cscd340$ valgrind ./a.out
==5224== Memcheck, a memory error detector
==5224== Copyright (C) 2002-2011, and GNU GPL'd, by Julian Seward et al.
==5224== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==5224== Command: ./a.out
==5224==
x: 0x7ff00043c
ptr: 0x7ff0003e0
val: 0x7ff000430
dptr: 0x7ff000428
array: 0x7ff000410
array[5]: 0x7ff000424
val2: 0x7ff000408
y:0x7ff000404
dptr2: 0x7ff0003f8
ptr2: 0x7ff0003f0
word: 0x7ff0003e8
word: 0x51f1040
==5224==
==5224== HEAP SUMMARY:
==5224==     in use at exit: 10 bytes in 1 blocks
==5224==   total heap usage: 1 allocs, 0 frees, 10 bytes allocated
==5224==
==5224== LEAK SUMMARY:
==5224==    definitely lost: 10 bytes in 1 blocks
==5224==    indirectly lost: 0 bytes in 0 blocks
```

```
==5224==      possibly lost: 0 bytes in 0 blocks
==5224==    still reachable: 0 bytes in 0 blocks
==5224==         suppressed: 0 bytes in 0 blocks
==5224== Rerun with --leak-check=full to see details of leaked memory
==5224==
==5224== For counts of detected and suppressed errors, rerun with: -v
==5224== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 2 from 2)
```