**Operating Systems Homework 03**

**Vijayasimha Bheemireddy**

**March 7th 2022**

1. First, write a simple program called null.c that creates a pointer to an integer, sets it to NULL, and then tries to dereference it. Compile this into an executable called null. What happens when you run this program?

A:

#include <stdio.h>

int main()

{

    int\* ptr = NULL;

    printf("%d", \*ptr);

}

Output:

Text

Description automatically generated

Explanation: Segmentation Fault occurs when I tried to dereference a null pointer.

2: Next, compile this program with symbol information included (with the -g flag). Doing so, let’s put more information into the executable, enabling the debugger to access more useful information about variable names and the like. Run the program under the debugger by typing gdb null and then, once gdb is running, typing run. What does gdb show you?

A:

Text

Description automatically generated

3: Finally, use the valgrind tool on this program. We’ll use the memcheck tool that is a part of valgrind to analyze what happens. Run this by typing in the following: valgrind --leak-check=yes null. What happens when you run this? Can you interpret the output from the tool?

A:

Explanation:

When I run this, it will she Segmentation fault as the output.

This output tells us that there is a memory error of null.c file. It tells us that the address we tried to access was not assigned (Access not within mapped region).

Yes, I can interpret the output from the tool.

Text

Description automatically generated

4: Write a simple program that allocates memory using malloc() but forgets to free it before exiting. What happens when this program runs? Can you use gdb to find any problems with it? How about valgrind (again with the --leak-check=yes flag)?

A:

Explanation:

Programs run correctly and When the program runs memory is allocated to heap.

No we cannot use gdb to find any problems.

According to valgrind we have leaked memory. We had one allocation but 0 frees.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int\* ptr = malloc(sizeof(int));

    \*ptr = 1;

    printf("%d", \*ptr)

}

Output:

Text

Description automatically generated

Gdb:

Text

Description automatically generated

Valgrind:

Text

Description automatically generated

5: Write a program that creates an array of integers called data of size 100 using malloc; then, set data[100] to zero. What happens when you run this program? What happens when you run this program using valgrind? Is the program correct?

A:

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int\* data = malloc(sizeof(int) \* 100);

    data[100] = 0;

}

Explanation:

Nothing happens when I try to run this program. I didn’t get any error when I run the program.

According to valgrind, I tried to perform an invalid write operation at line 7 of task5.c file.

Furthermore, it tells us that we had one alloc operation that allocated 400 bytes, but we did not have any free operation which means we have memory leakage. So, the program is not correct.

There are 2 errors occurred from 2 contexts.

A black screen with white text

Description automatically generated with low confidence

6: Create a program that allocates an array of integers (as above), frees them, and then tries to print the value of one of the elements of the array. Does the program run? What happens when you use valgrind on it?

A:

Explanation: Yes, The program printed '0' upon running it and ended without errors.

According to valgrind, we have an invalid read operation at line 9 in task6.c file (When we try to printf the value). Further valgrind tells us that we freed all our allocated memory. When I use valgrind on the program all heap blocks are freed and no leaks are possible.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int\* data = malloc(sizeof(int) \* 100);

    free(data);

    printf("%d", data[0]);

}

Output :

0

Valgrind:

Text

Description automatically generated

7: Now pass a funny value to free (e.g., a pointer in the middle of the array you allocated above). What happens? Do you need tools to find this type of problem?

A:

Explanation:

We had error when we tried to execute this program without any extra information tools.

Terminal Output: free(): invalid pointer Aborted (core dumped)

In order to solve the errors we need gdb and valgrind tools for this type of problem.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int\* data = malloc(sizeof(int) \* 100);

    free(&data[10]);

printf(“%d \n”,data[11]);

}

Output:

Text

Description automatically generated

Valgrind:

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated