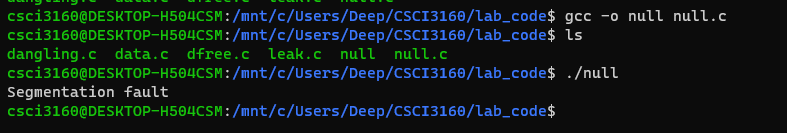
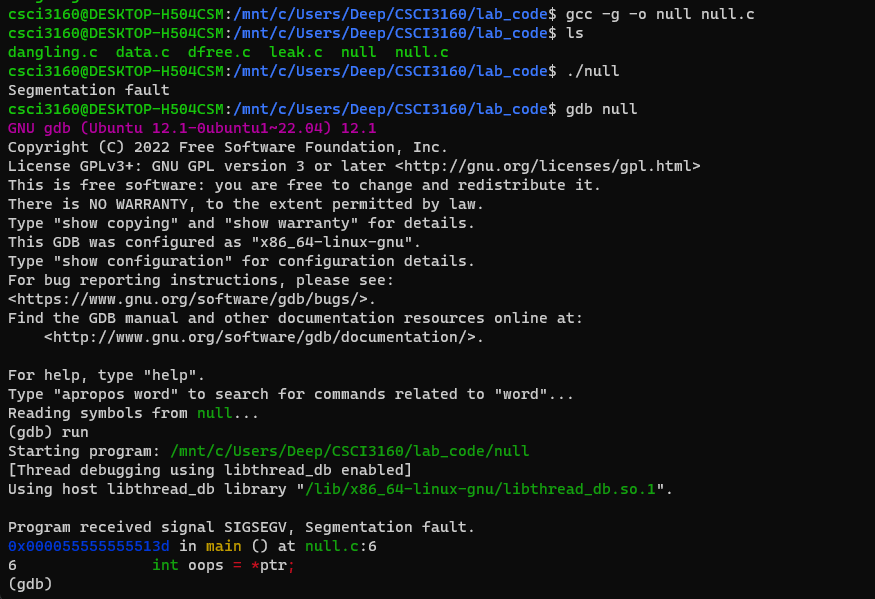
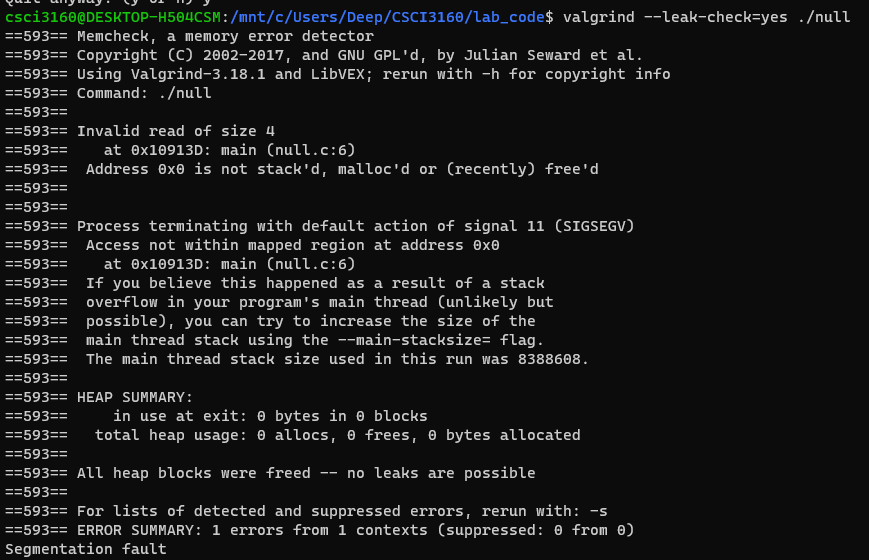
**Lab Procedure CSCI-3160**



Null.c Compiled and Ran



Null.c GDB Debugger



Null.c Valgrind Profiler

1. Write a program called null.c that creates a pointer to an integer, sets it to NULL (literally, NULL), and then tries to dereference that pointer. Compile this program (using gcc) into an executable called null. Take a screenshot of the output and include it in your lab report.

• TODO: What happens when you run this program?

Gets a segmentation fault error and invalid read of size 4.

A screenshot of a computer program

Description automatically generated

Leak.c Compiled. Ran, and Debugger

A computer screen with white text

Description automatically generated

Leak.c Valgrind Profiler

Write a program called leak.c that allocates memory using malloc() but does not call free() before  
exiting. Compile this program to leak. What happens when this program runs? When you compile with  
debugging symbols, does gdb detect any issue? How about valgrind ($ valgrind --leak-check=yes  
./leak)? Take screenshots of the output from each step and include it in your lab report.

• TODO: What kind of error does forget to free a memory allocation cause?

Caused a memory leak error. Allocated memory but didn’t free it after use.

A computer screen shot of a computer code

Description automatically generated

Data.c Compiled, Ran, and Debugger

A screenshot of a computer screen

Description automatically generated

Data.c Valgrind Profiler

Write a program called data.c that creates an array of integers (called data) of size 100 using malloc();  
then, set data[100] to 0. Compile this program to data. What happens when this program runs? When you  
compile with debugging symbols, does gdb detect any issue? How about valgrind? Take screenshots of  
the output from each step and include it in your lab report.

• TODO: What kind of error does the data program cause?

We wrote past the bounds of the array in memory. It’s not a problem here because this was a simple example; however, if something in memory was stored right next to this array, our data would have overwritten it.

A computer screen with white text and green text

Description automatically generated

Dangling.c Compiled, Ran, and Debugger

A screenshot of a computer program

Description automatically generated

Dangling.c Valgrind Profiler

Write a program named dangling.c that also creates an array of 100 integers (called data) using malloc(), frees them (using free()), then tries to print the value of one of the elements of the array.  
Compile this program to dangling. What happens when this program runs? When you compile with debugging symbols, does gdb detect any issue? How about valgrind? Take screenshots of the output from each step and include it in your lab report.

• TODO: What kind of error does the dangling program cause?

Trying to access a dead pointer.

A computer screen with white text

Description automatically generated

Dfree.c Compiling – Throwing Error

Write a program named dfree.c that allocates and frees an array of 100 integers (called data), but passes data[50] to free(). Compile this program to dfree. What happens when this program runs? When you compile with debugging symbols, does gdb detect any issue? How about valgrind? Take screenshots of  
the output from each step and include it in your lab report.

• TODO: What kind of error does the dfree program cause?

Wrong data type, compiler stopped the error before compiling.