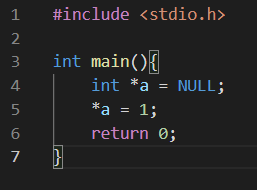
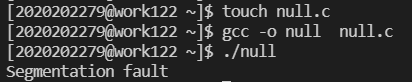
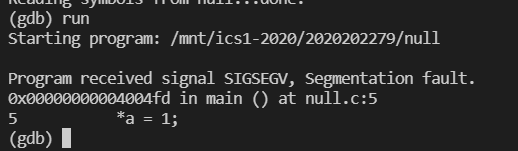
1.按照要求，编写如下null.c程序



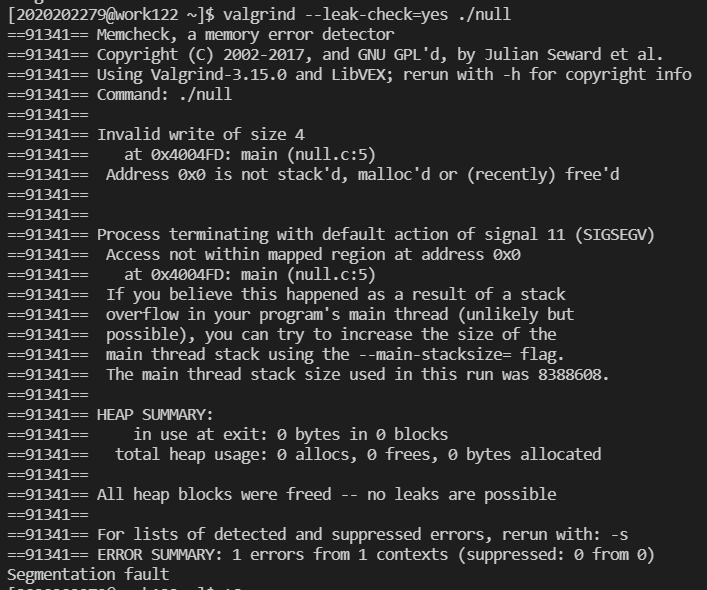
编译运行得到报错：Segmentation fault，如下：



2. gdb展示的信息：程序接受SIGSEGV信号终止



3.按要求输入命令，得到如下输出：



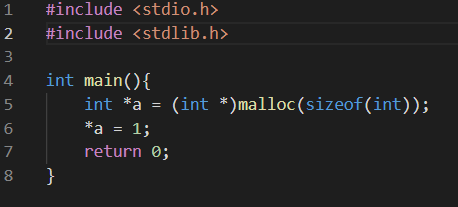
其中，Invalid write of size 4指出了程序的错误：在null.c程序的第五行，尝试为指针a指向的空间赋值，但a是空指针，对应的地址0x0不是可访问（写入）的空间，因此产生错误。

下一段则说明程序收到了信号11（SIGSEGV）而终止，因为访问不在地址0x0的映射区域内。并给出针对可能出现的栈溢出情况给出提醒和建议。

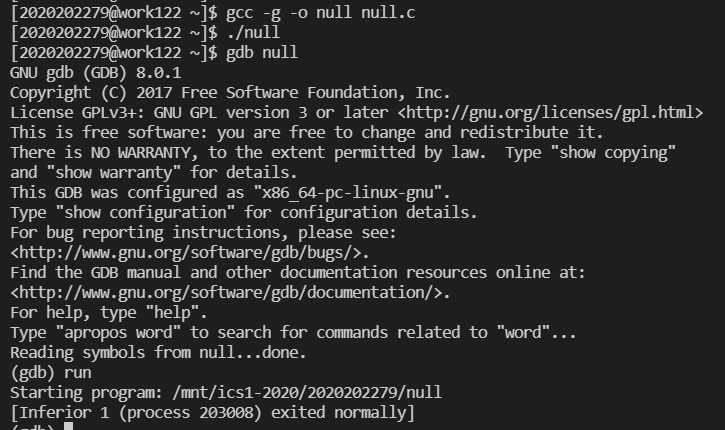
下一段给出了堆区的总结，包括使用情况（多少allocs、多少frees等）和释放情况。

最后则是错误总结。

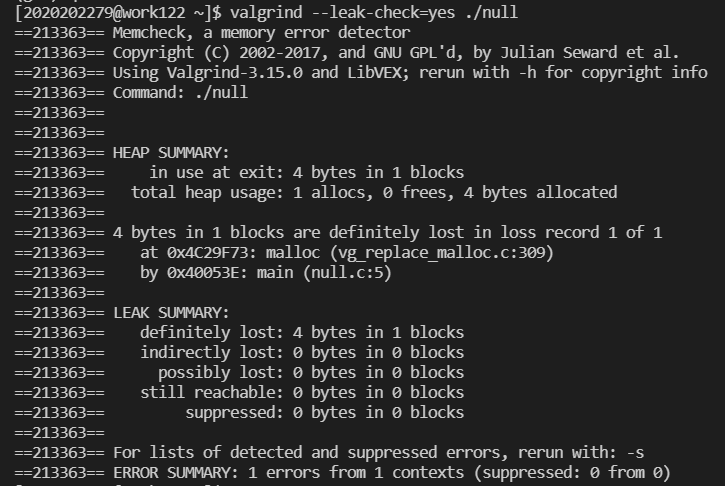
4.按要求改写程序，malloc但不free

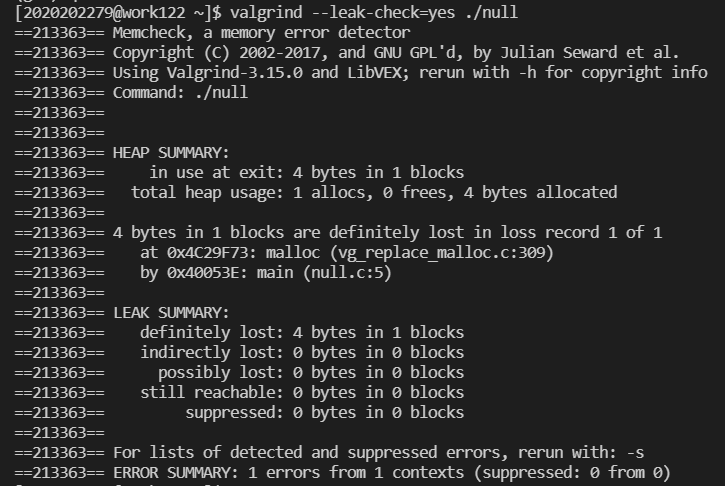
****

编译运行没有报错，gdb中进程也正常退出，所以不能用gdb找到问题

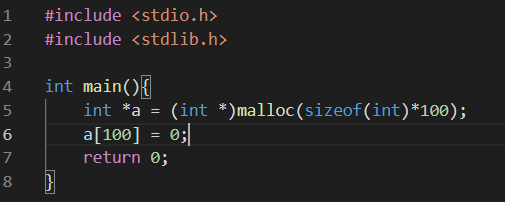


而valgrind的输出则不同，在对堆区进行总结时，指出进行了1个alloc和0个free，并给出了对应的未释放的malloc指令的位置和所在函数。此外，还对内存泄露的总情况进行了总结和描述，指出了程序的错误。

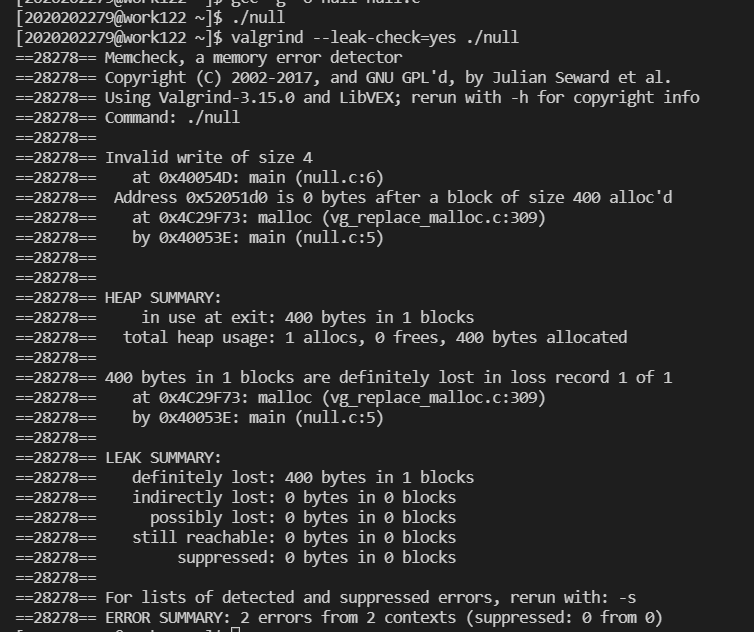




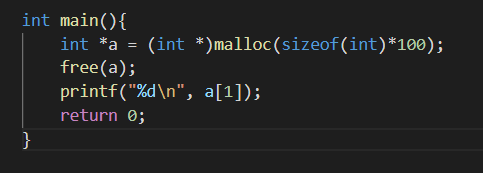
5.



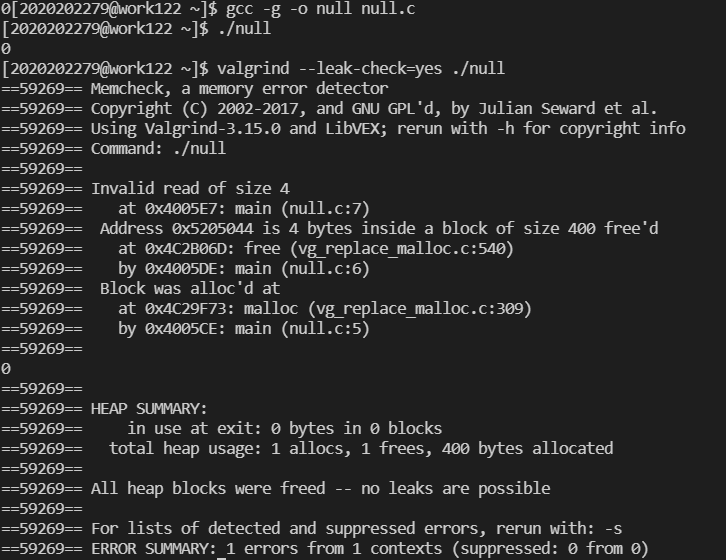
编译运行不报错，valgrind则提示访问a[100]是不合法的，并且未进行free导致内存泄漏。因此，这个程序是不正确的，它访问了超出malloc分配空间的a[100]，而且未释放分配的空间。



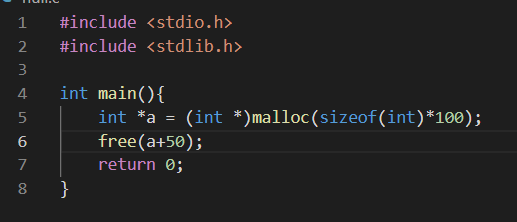
6.



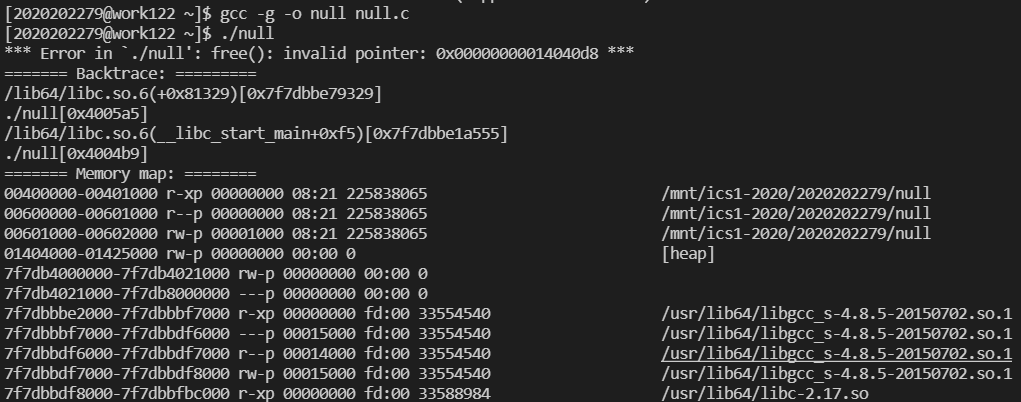
该程序可以运行并且输出0，但是valgrind工具则报错，显示它进行了非法读取，因为对应的空间已经free释放。不过此次由于申请的空间已经释放，所以没有内存泄漏。



7.



编译运行直接报错，所以不需要gdb或valgrind工具即可发现错误，即不能用指向数组中间值的指针来进行释放。



8.经过实验，用realloc为向量扩容的方法只适合动态数组（如malloc分配的数组），而不适合静态数组。并且由于realloc在其后空间不足时，需要为原数组分配新的空间并将其复制过去，所以在数据规模较大或需要多次扩容时，该种向量的性能会有所下降。

与链表相比，这种向量要求存储空间必须是连续的，而链表则对空间的连续性没有要求，因此不需要对原数组进行复制。