Pre-lab questions

1.1 The purpose of the “man 1 man” command, is to find and display online documentation pages. It is also a utility that provides an interface to the online reference manual.

1.2 The difference between section 1 (user commands) and section 3 (library functions).

> Section 1 of the manual describes user commands and tools, for example, file manipulation tools, shells, compilers, web browsers, file and image viewers and editors, and so on.
>
> All commands yield a status value on termination. This value can be tested to see whether the command is completed successfully. A zero-exit status is conventionally used to indicate success, and a nonzero status means that the command was unsuccessful. A nonzero exit status can be in the range 1 to 255, and some commands use different nonzero status values to indicate the reason why the command failed.
>
> Section 3 of the manual describes all library functions excluding the library functions described in section 2, which implement system calls.
>
> Many of the functions described in the section are part of the Standard C Library. Some functions are part of other libraries in which case the manual page will indicate the linker option needed to link against the required library.

1.3 The difference between write(1) and fwrite(3)

- write(1) - Write to a file descriptor. Write() writes up to count byte (1) from the buffer pointed to the file referred to by the file descriptor fd. On success, the number of bytes written is returned (zero indicates nothing was written). On error, -1 is returned, and errno is set appropriately.
- fwrite(3) - Binary stream input / output. The function fwrite() writes nmemb elements of data, each size bytes long, to the stream pointed to by stream, obtaining them from the location given by ptr. On success, fwrite() return the number of items read or written. This number equals the number of bytes transferred only when size is 1. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).

1.4 The malloc functions returns a pointer to the allocated memory that is suitably aligned for any kindd of variable. On error, these functions return NULL. NULL may also be returned by a successful call to malloc() with a size of zero.

2.

```
1 #ifndef ADD_H
2 #define ADD_H
3
```

```
4 #include <iostream>
5
6 int add(int, int);
7 float add(float, float);
8
9 #endif
~
```

```
1 #include "add.h"
2
3 int add(int a, int b)
4 {
5     return int(a + b);
6 }
7
8 float add(float a, float b)
9 {
10     return float(a + b);
11 }
12
```

Procedure

Symbols and polymorphism

1.

```
0000000000000018 T add(float, float)
0000000000000000 T add(int, int)
```

2.

```
0000000000000000 T add
0000000000000018 T add(float, float)
```

3.

After putting the code from part 2 into a ".c" file we notice that the float version of the add function no longer works because C does not have operator overloading.

```
c-math.c:8:7: error: conflicting types for ‘add’; have
‘float(float,  float)’
    8 | float add(float a, float b)
```

```
      |       ^~~
c-math.c:3:5: note: previous definition of ‘add’ with type
‘int(int,  int)’
    3 | int add(int a, int b)
      |     ^~~
```

4.

C

```
0000000000000000 <add>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   89 7d fc                mov    %edi,-0x4(%rbp)
   b:   89 75 f8                mov    %esi,-0x8(%rbp)
   e:   8b 55 fc                mov    -0x4(%rbp),%edx
  11:   8b 45 f8                mov    -0x8(%rbp),%eax
  14:   01 d0                   add    %edx,%eax
  16:   5d                      pop    %rbp
  17:   c3                      retq
```

CC

```
0000000000000000 <add>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   89 7d fc                mov    %edi,-0x4(%rbp)
   b:   89 75 f8                mov    %esi,-0x8(%rbp)
   e:   8b 55 fc                mov    -0x4(%rbp),%edx
  11:   8b 45 f8                mov    -0x8(%rbp),%eax
  14:   01 d0                   add    %edx,%eax
  16:   5d                      pop    %rbp
  17:   c3                      retq

0000000000000018 <_Z3addff>:
  18:   f3 0f 1e fa             endbr64
  1c:   55                      push   %rbp
  1d:   48 89 e5                mov    %rsp,%rbp
  20:   f3 0f 11 45 fc          movss  %xmm0,-0x4(%rbp)
```

```
  25:   f3 0f 11 4d f8          movss  %xmm1,-0x8(%rbp)
  2a:   f3 0f 10 45 fc          movss  -0x4(%rbp),%xmm0
  2f:   f3 0f 58 45 f8          addss  -0x8(%rbp),%xmm0
  34:   5d                      pop    %rbp
  35:   c3                      retq
```

C-Style Strings

1.

Breakpoint 1, main () at main.c:8

warning: Source file is more recent than executable.

```
8           fp = fopen("file.txt", "w");
(gdb) print &str
$1 = (char (*)[23]) 0x7fffffffd510
```

2.

```
Breakpoint 1, main () at main.c:22
22          return 0;
(gdb) print length
$1 = 3
(gdb) x/8x name
0x555555556004:
0x00534443      0x3b031b01      0x00000034      0x00000005
0x555555556014:
0xfffff018      0x00000068      0xfffff038      0x00000090
Little-endian so read from right to left
43 = C
44 = D
53 = S
00 = NULL

My favorite integer is : 10
floating-point : 24.000000, Hex : 0x0018, Pointer: 0x7ffdd4eda304.
```

3.

```
Breakpoint 1, main () at main.c:23
23          return 0;
(gdb) print name
$1 = "Char"
(gdb) x/8x name
0x7fffffffd524:
0x72616843      0x0073656c      0xaa426c7a      0x00000001

0x7fffffffd534:
0x00000000      0xf7c29d90      0x00007fff      0x00000000
```

I/O with libc

```
My fauourite integer is : 10
floating-point : 24.000000, Hex : 0x0018, Pointer: 0x7ffe13990ba4.
My name is 'Charles Smith', which is 14 characters long, and it is
located at address 0x7ffe13990bb2.
```

Memory Management

1.

```
Address = 0x5561105b52a0
```

2.

Without fsanitize=address flag

```
Address = 0x55703e12f2a0
Address = 0x55de8eafb2a0
```

When ran without the fsanitize=address flag the address memory changes each time you execute the output file.

With fsanitize=address flag

```
Address = 0x602000000010
```

When ran with the fsanitize=address flag the address does not change no matter how much times you execute the output file.

3.

```
(malloc) Address = 0x559abf80f2a0
(calloc) Address = 0x559abf810250
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Registered pretty printers for UE classes
Reading symbols from ./memMan2...
(gdb) break 12
Breakpoint 1 at 0x1271: file memManagement2.c, line 12.
(gdb) run
Starting program: /home/charlessmith/Documents/Term8/ECE8400-RealTime-
Operating-Systems/Labs/Lab1/memMan2
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-
gnu/libthread_db.so.1".
(malloc) Address = 0x621000000100

Breakpoint 1, main () at memManagement2.c:12
12          printf("(calloc) Address = %p \n", num1);
(gdb) print/u *num0@1000
$1 = {3200171710 <repeats 1000 times>}
(gdb) print/u *num1@1000
$2 = {0 <repeats 1000 times>}
(gdb)
```

As seen in the image above there are differences between the malloc and calloc pointers. Malloc seems to store its container size in the pointer where calloc always sets it to 0, calloc also has an additional attribute called arraySize which are the number of blocks to store integers in this case.

4.

Without fsanitize=address

```
Breakpoint 1, main () at memManagement4.c:7
7           printf("Address = %p \n", num0);
(gdb) x/100 num0
0x5555555592a0: 0       0       0       0
0x5555555592b0: 0       0       0       0
0x5555555592c0: 0       0       0       0
0x5555555592d0: 0       0       0       0
0x5555555592e0: 0       0       0       0
0x5555555592f0: 0       0       0       0
```

```
0x555555559300: 0       0       134401  0
0x555555559310: 0       0       0       0
0x555555559320: 0       0       0       0
0x555555559330: 0       0       0       0
0x555555559340: 0       0       0       0
0x555555559350: 0       0       0       0
0x555555559360: 0       0       0       0
0x555555559370: 0       0       0       0
0x555555559380: 0       0       0       0
0x555555559390: 0       0       0       0
0x5555555593a0: 0       0       0       0
0x5555555593b0: 0       0       0       0
0x5555555593c0: 0       0       0       0
0x5555555593d0: 0       0       0       0
0x5555555593e0: 0       0       0       0
0x5555555593f0: 0       0       0       0
0x555555559400: 0       0       0       0
0x555555559410: 0       0       0       0
0x555555559420: 0       0       0       0
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) continue
Continuing.
Address = 0x5555555592a0
Address = 0x5555555592a0

Breakpoint 2, main () at memManagement4.c:12
12          printf("Break");
(gdb) x/100 num0
0x5555555592a0: 1431655769      5       -316035110      2118954817
0x5555555592b0: 0       0       0       0
0x5555555592c0: 0       0       0       0
0x5555555592d0: 0       0       0       0
0x5555555592e0: 0       0       0       0
0x5555555592f0: 0       0       0       0
0x555555559300: 0       0       1041    0
0x555555559310:
1919181889      544437093       2016419901      892679477
0x555555559320: 892679477       811676217       2592    0
0x555555559330: 0       0       0       0
0x555555559340: 0       0       0       0
0x555555559350: 0       0       0       0
0x555555559360: 0       0       0       0
0x555555559370: 0       0       0       0
0x555555559380: 0       0       0       0
0x555555559390: 0       0       0       0
```

```
0x5555555593a0: 0       0       0       0
0x5555555593b0: 0       0       0       0
0x5555555593c0: 0       0       0       0
0x5555555593d0: 0       0       0       0
0x5555555593e0: 0       0       0       0
0x5555555593f0: 0       0       0       0
0x555555559400: 0       0       0       0
0x555555559410: 0       0       0       0
0x555555559420: 0       0       0       0
```

With fsanitize=address

```
Breakpoint 1, main () at memManagement4.c:7
7           printf("Address = %p \n", num0);
(gdb) x/100 num0
0x60b0000000f0: 0       0       0       0
0x60b000000100: 0       0       0       0
0x60b000000110: 0       0       0       0
0x60b000000120: 0       0       0       0
0x60b000000130: 0       0       0       0
0x60b000000140: 0       0       0       0
0x60b000000150: 0       0       0       0
0x60b000000160: 0       0       0       0
0x60b000000170: 0       0       0       0
0x60b000000180: 0       0       0       0
0x60b000000190: 0       0       0       0
0x60b0000001a0: 0       0       0       0
0x60b0000001b0: 0       0       0       0
0x60b0000001c0: 0       0       0       0
0x60b0000001d0: 0       0       0       0
0x60b0000001e0: 0       0       0       0
0x60b0000001f0: 0       0       0       0
0x60b000000200: 0       0       0       0
0x60b000000210: 0       0       0       0
0x60b000000220: 0       0       0       0
0x60b000000230: 0       0       0       0
0x60b000000240: 0       0       0       0
0x60b000000250: 0       0       0       0
0x60b000000260: 0       0       0       0
0x60b000000270: 0       0       0       0
(gdb) continue
Continuing.
Address = 0x60b0000000f0
Address = 0x60b0000000f0

Breakpoint 2, main () at memManagement4.c:12
12          printf("Break");
```

```
(gdb) x/100 num0
0x60b0000000f0: 1       0       0       0
0x60b000000100: 0       0       0       0
0x60b000000110: 0       0       0       0
0x60b000000120: 0       0       0       0
0x60b000000130: 0       0       0       0
0x60b000000140: 0       0       0       0
0x60b000000150: 0       0       0       0
0x60b000000160: 0       0       0       0
0x60b000000170: 0       0       0       0
0x60b000000180: 0       0       0       0
0x60b000000190: 0       0       0       0
0x60b0000001a0: 0       0       0       0
0x60b0000001b0: 0       0       0       0
0x60b0000001c0: 0       0       0       0
0x60b0000001d0: 0       0       0       0
0x60b0000001e0: 0       0       0       0
0x60b0000001f0: 0       0       0       0
0x60b000000200: 0       0       0       0
0x60b000000210: 0       0       0       0
0x60b000000220: 0       0       0       0
0x60b000000230: 0       0       0       0
0x60b000000240: 0       0       0       0
0x60b000000250: 0       0       0       0
0x60b000000260: 0       0       0       0
0x60b000000270: 0       0       0       0
```

As seen

As seen above the pointer array address does not change address but after freeing the pointer a value of 1 at address `0x60b0000000f0` whereas before it is all initialize to 0. Which means that it is going back to random garbage values.

5.

```
Address = 0x60b0000000f0
Address = 0x60b0000000f0
=================================================================
==33136==ERROR: AddressSanitizer: attempting double-free on
0x60b0000000f0 in thread T0:
    #0 0x7fbadceb4517 in __interceptor_free
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:127
    #1 0x55d8d4a0f275 in main
/home/charlessmith/Documents/Term8/ECE8400-RealTime-Operating-
Systems/Labs/Lab1/memManagement4.c:11
    #2 0x7fbadca29d8f in __libc_start_call_main
../sysdeps/nptl/libc_start_call_main.h:58
```

```
    #3 0x7fbadca29e3f in __libc_start_main_impl ../csu/libc-
start.c:392
    #4 0x55d8d4a0f144 in _start
(/home/charlessmith/Documents/Term8/ECE8400-RealTime-Operating-
Systems/Labs/Lab1/memMan4+0x1144)

0x60b0000000f0 is located 0 bytes inside of 100-byte region
[0x60b0000000f0,0x60b000000154)
freed by thread T0 here:
    #0 0x7fbadceb4517 in __interceptor_free
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:127
    #1 0x55d8d4a0f24e in main
/home/charlessmith/Documents/Term8/ECE8400-RealTime-Operating-
Systems/Labs/Lab1/memManagement4.c:9
    #2 0x7fbadca29d8f in __libc_start_call_main
../sysdeps/nptl/libc_start_call_main.h:58

previously allocated by thread T0 here:
    #0 0x7fbadceb4a37 in __interceptor_calloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:154
    #1 0x55d8d4a0f223 in main
/home/charlessmith/Documents/Term8/ECE8400-RealTime-Operating-
Systems/Labs/Lab1/memManagement4.c:6
    #2 0x7fbadca29d8f in __libc_start_call_main
../sysdeps/nptl/libc_start_call_main.h:58

SUMMARY: AddressSanitizer: double-free
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:127 in
__interceptor_free
```

When we call free(3) on an already deallocated array the program aborts due to a core dump.

6.

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Registered pretty printers for UE classes
Reading symbols from ./memMan4...
(gdb)
(gdb) break 16\
Breakpoint 1 at 0x1224: file memManagement4.c, line 16.
(gdb) run
Starting program: /home/charlessmith/Documents/Term8/ECE8400-RealTime-
Operating-Systems/Labs/Lab1/memMan4
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-
gnu/libthread_db.so.1".
```

```
Breakpoint 1, main () at memManagement4.c:16
16          return 0;
(gdb) x/1 num1
0x0:    Cannot access memory at address 0x0
```

When calling free(3) on a NULL pointer the program continues as if everything is fine, no errors no warnings.

The value of the pointer can no longer be accesseed.

Macros

1.

```
aaron@aaron-X541UAK:~/School/Term8/OS/labs2023/1$ cc -E
macroQuestions.c
# 1 "macroQuestions.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "macroQuestions.c"
```

After pre-processing the file the file did not contain the string FOO.

2.

```
aaron@aaron-X541UAK:~/School/Term8/OS/labs2023/1$ cc -E
macroQuestions.c
# 1 "macroQuestions.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "macroQuestions.c"
```

```
void bar()
{
    for(int i = 0; i < 2; i++)
    {
        "this is the definition of foo"+i;
```

```
    }
}
```

The first 7 lines of code remained similar, the code with the function using FOO actually displayed all of the function with the macro FOO being replaced by the string contained by FOO.

3

```
# 0 "macro1.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "macro1.c"


# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 392 "/usr/include/features.h" 3 4
# 1 "/usr/include/features-time64.h" 1 3 4
# 20 "/usr/include/features-time64.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 21 "/usr/include/features-time64.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 1 3 4
# 19 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 20 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 2 3 4
# 22 "/usr/include/features-time64.h" 2 3 4
# 393 "/usr/include/features.h" 2 3 4
# 486 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 559 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 560 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1 3 4
# 561 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 487 "/usr/include/features.h" 2 3 4
# 510 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
```

```
# 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
# 511 "/usr/include/features.h" 2 3 4
# 34 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4




# 1 "/usr/lib/gcc/x86_64-linux-gnu/11/include/stddef.h" 1 3 4
# 209 "/usr/lib/gcc/x86_64-linux-gnu/11/include/stddef.h" 3 4

# 209 "/usr/lib/gcc/x86_64-linux-gnu/11/include/stddef.h" 3 4
typedef long unsigned int size_t;
# 34 "/usr/include/stdio.h" 2 3 4


# 1 "/usr/lib/gcc/x86_64-linux-gnu/11/include/stdarg.h" 1 3 4
# 40 "/usr/lib/gcc/x86_64-linux-gnu/11/include/stdarg.h" 3 4
typedef __builtin_va_list __gnuc_va_list;
# 37 "/usr/include/stdio.h" 2 3 4

# 1 "/usr/include/x86_64-linux-gnu/bits/types.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 28 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 1 3 4
# 19 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 20 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 2 3 4
# 29 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;


typedef signed char __int8_t;
typedef unsigned char __uint8_t;
typedef signed short int __int16_t;
typedef unsigned short int __uint16_t;
typedef signed int __int32_t;
typedef unsigned int __uint32_t;

typedef signed long int __int64_t;
typedef unsigned long int __uint64_t;
```

```
typedef __int8_t __int_least8_t;
typedef __uint8_t __uint_least8_t;
typedef __int16_t __int_least16_t;
typedef __uint16_t __uint_least16_t;
typedef __int32_t __int_least32_t;
typedef __uint32_t __uint_least32_t;
typedef __int64_t __int_least64_t;
typedef __uint64_t __uint_least64_t;



typedef long int __quad_t;
typedef unsigned long int __u_quad_t;









typedef long int __intmax_t;
typedef unsigned long int __uintmax_t;
# 141 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/typesizes.h" 1 3 4
# 142 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/time64.h" 1 3 4
# 143 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned long int __dev_t;
typedef unsigned int __uid_t;
typedef unsigned int __gid_t;
typedef unsigned long int __ino_t;
typedef unsigned long int __ino64_t;
typedef unsigned int __mode_t;
typedef unsigned long int __nlink_t;
typedef long int __off_t;
typedef long int __off64_t;
typedef int __pid_t;
typedef struct { int __val[2]; } __fsid_t;
typedef long int __clock_t;
typedef unsigned long int __rlim_t;
```

```
typedef unsigned long int __rlim64_t;
typedef unsigned int __id_t;
typedef long int __time_t;
typedef unsigned int __useconds_t;
typedef long int __suseconds_t;
typedef long int __suseconds64_t;

typedef int __daddr_t;
typedef int __key_t;


typedef int __clockid_t;


typedef void * __timer_t;


typedef long int __blksize_t;




typedef long int __blkcnt_t;
typedef long int __blkcnt64_t;


typedef unsigned long int __fsblkcnt_t;
typedef unsigned long int __fsblkcnt64_t;


typedef unsigned long int __fsfilcnt_t;
typedef unsigned long int __fsfilcnt64_t;


typedef long int __fsword_t;

typedef long int __ssize_t;


typedef long int __syscall_slong_t;

typedef unsigned long int __syscall_ulong_t;



typedef __off64_t __loff_t;
typedef char *__caddr_t;
```

```
typedef long int __intptr_t;


typedef unsigned int __socklen_t;




typedef int __sig_atomic_t;
# 39 "/usr/include/stdio.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types/__fpos_t.h" 1 3 4




# 1 "/usr/include/x86_64-linux-gnu/bits/types/__mbstate_t.h" 1 3 4
# 13 "/usr/include/x86_64-linux-gnu/bits/types/__mbstate_t.h" 3 4
typedef struct
{
  int __count;
  union
  {
    unsigned int __wch;
    char __wchb[4];
  } __value;
} __mbstate_t;
# 6 "/usr/include/x86_64-linux-gnu/bits/types/__fpos_t.h" 2 3 4




typedef struct _G_fpos_t
{
  __off_t __pos;
  __mbstate_t __state;
} __fpos_t;
# 40 "/usr/include/stdio.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types/__fpos64_t.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/bits/types/__fpos64_t.h" 3 4
typedef struct _G_fpos64_t
{
  __off64_t __pos;
  __mbstate_t __state;
} __fpos64_t;
# 41 "/usr/include/stdio.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types/__FILE.h" 1 3 4
```

```
struct _IO_FILE;
typedef struct _IO_FILE __FILE;
# 42 "/usr/include/stdio.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types/FILE.h" 1 3 4



struct _IO_FILE;


typedef struct _IO_FILE FILE;
# 43 "/usr/include/stdio.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types/struct_FILE.h" 1 3 4
# 35 "/usr/include/x86_64-linux-gnu/bits/types/struct_FILE.h" 3 4
struct _IO_FILE;
struct _IO_marker;
struct _IO_codecvt;
struct _IO_wide_data;




typedef void _IO_lock_t;




struct _IO_FILE
{
  int _flags;


  char *_IO_read_ptr;
  char *_IO_read_end;
  char *_IO_read_base;
  char *_IO_write_base;
  char *_IO_write_ptr;
  char *_IO_write_end;
  char *_IO_buf_base;
  char *_IO_buf_end;


  char *_IO_save_base;
  char *_IO_backup_base;
  char *_IO_save_end;
```

```
  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
  int _flags2;
  __off_t _old_offset;


  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];

  _IO_lock_t *_lock;









  __off64_t _offset;

  struct _IO_codecvt *_codecvt;
  struct _IO_wide_data *_wide_data;
  struct _IO_FILE *_freeres_list;
  void *_freeres_buf;
  size_t __pad5;
  int _mode;

  char _unused2[15 * sizeof (int) - 4 * sizeof (void *) - sizeof
(size_t)];
};
# 44 "/usr/include/stdio.h" 2 3 4
# 52 "/usr/include/stdio.h" 3 4
typedef __gnuc_va_list va_list;
# 63 "/usr/include/stdio.h" 3 4
typedef __off_t off_t;
# 77 "/usr/include/stdio.h" 3 4
typedef __ssize_t ssize_t;







typedef __fpos_t fpos_t;
```

```
# 133 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/stdio_lim.h" 1 3 4
# 134 "/usr/include/stdio.h" 2 3 4
# 143 "/usr/include/stdio.h" 3 4
extern FILE *stdin;
extern FILE *stdout;
extern FILE *stderr;




extern int remove (const char *__filename) __attribute__ ((__nothrow__
, __leaf__));

extern int rename (const char *__old, const char *__new) __attribute__
((__nothrow__ , __leaf__));



extern int renameat (int __oldfd, const char *__old, int __newfd,
       const char *__new) __attribute__ ((__nothrow__ , __leaf__));
# 178 "/usr/include/stdio.h" 3 4
extern int fclose (FILE *__stream);
# 188 "/usr/include/stdio.h" 3 4
extern FILE *tmpfile (void)
  __attribute__ ((__malloc__)) __attribute__ ((__malloc__ (fclose,
1))) ;
# 205 "/usr/include/stdio.h" 3 4
extern char *tmpnam (char[20]) __attribute__ ((__nothrow__ ,
__leaf__)) ;




extern char *tmpnam_r (char __s[20]) __attribute__ ((__nothrow__ ,
__leaf__)) ;
# 222 "/usr/include/stdio.h" 3 4
extern char *tempnam (const char *__dir, const char *__pfx)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__
((__malloc__)) __attribute__ ((__malloc__ (__builtin_free, 1)));
```

```
extern int fflush (FILE *__stream);
# 239 "/usr/include/stdio.h" 3 4
extern int fflush_unlocked (FILE *__stream);
# 258 "/usr/include/stdio.h" 3 4
extern FILE *fopen (const char *__restrict __filename,
      const char *__restrict __modes)
  __attribute__ ((__malloc__)) __attribute__ ((__malloc__ (fclose,
1))) ;



extern FILE *freopen (const char *__restrict __filename,
        const char *__restrict __modes,
        FILE *__restrict __stream) ;
# 293 "/usr/include/stdio.h" 3 4
extern FILE *fdopen (int __fd, const char *__modes) __attribute__
((__nothrow__ , __leaf__))
  __attribute__ ((__malloc__)) __attribute__ ((__malloc__ (fclose,
1))) ;
# 308 "/usr/include/stdio.h" 3 4
extern FILE *fmemopen (void *__s, size_t __len, const char *__modes)
  __attribute__ ((__nothrow__ , __leaf__)) __attribute__
((__malloc__)) __attribute__ ((__malloc__ (fclose, 1))) ;



extern FILE *open_memstream (char **__bufloc, size_t *__sizeloc)
__attribute__ ((__nothrow__ , __leaf__))
  __attribute__ ((__malloc__)) __attribute__ ((__malloc__ (fclose,
1))) ;
# 328 "/usr/include/stdio.h" 3 4
extern void setbuf (FILE *__restrict __stream, char *__restrict __buf)
__attribute__ ((__nothrow__ , __leaf__));



extern int setvbuf (FILE *__restrict __stream, char *__restrict __buf,
      int __modes, size_t __n) __attribute__ ((__nothrow__ ,
__leaf__));



extern void setbuffer (FILE *__restrict __stream, char *__restrict
__buf,
        size_t __size) __attribute__ ((__nothrow__ , __leaf__));
```

```
extern void setlinebuf (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__));
```

```
extern int fprintf (FILE *__restrict __stream,
      const char *__restrict __format, ...);
```

```
extern int printf (const char *__restrict __format, ...);

extern int sprintf (char *__restrict __s,
      const char *__restrict __format, ...) __attribute__
((__nothrow__));
```

```
extern int vfprintf (FILE *__restrict __s, const char *__restrict
__format,
       __gnuc_va_list __arg);
```

```
extern int vprintf (const char *__restrict __format, __gnuc_va_list
__arg);

extern int vsprintf (char *__restrict __s, const char *__restrict
__format,
       __gnuc_va_list __arg) __attribute__ ((__nothrow__));
```

```
extern int snprintf (char *__restrict __s, size_t __maxlen,
       const char *__restrict __format, ...)
     __attribute__ ((__nothrow__)) __attribute__ ((__format__
(__printf__, 3, 4)));
```

```
extern int vsnprintf (char *__restrict __s, size_t __maxlen,
        const char *__restrict __format, __gnuc_va_list __arg)
     __attribute__ ((__nothrow__)) __attribute__ ((__format__
(__printf__, 3, 0)));
# 403 "/usr/include/stdio.h" 3 4
extern int vdprintf (int __fd, const char *__restrict __fmt,
       __gnuc_va_list __arg)
     __attribute__ ((__format__ (__printf__, 2, 0)));
extern int dprintf (int __fd, const char *__restrict __fmt, ...)
     __attribute__ ((__format__ (__printf__, 2, 3)));







extern int fscanf (FILE *__restrict __stream,
     const char *__restrict __format, ...) ;




extern int scanf (const char *__restrict __format, ...) ;

extern int sscanf (const char *__restrict __s,
     const char *__restrict __format, ...) __attribute__ ((__nothrow__
, __leaf__));






# 1 "/usr/include/x86_64-linux-gnu/bits/floatn.h" 1 3 4
# 119 "/usr/include/x86_64-linux-gnu/bits/floatn.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/floatn-common.h" 1 3 4
# 24 "/usr/include/x86_64-linux-gnu/bits/floatn-common.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1 3 4
# 25 "/usr/include/x86_64-linux-gnu/bits/floatn-common.h" 2 3 4
# 120 "/usr/include/x86_64-linux-gnu/bits/floatn.h" 2 3 4
# 431 "/usr/include/stdio.h" 2 3 4



extern int fscanf (FILE *__restrict __stream, const char *__restrict
__format, ...) __asm__ ("" "__isoc99_fscanf")

                               ;
```

```
extern int scanf (const char *__restrict __format, ...) __asm__ (""
"__isoc99_scanf")
                               ;
extern int sscanf (const char *__restrict __s, const char *__restrict
__format, ...) __asm__ ("" "__isoc99_sscanf") __attribute__
((__nothrow__ , __leaf__))

                       ;
# 459 "/usr/include/stdio.h" 3 4
extern int vfscanf (FILE *__restrict __s, const char *__restrict
__format,
      __gnuc_va_list __arg)
     __attribute__ ((__format__ (__scanf__, 2, 0))) ;




extern int vscanf (const char *__restrict __format, __gnuc_va_list
__arg)
     __attribute__ ((__format__ (__scanf__, 1, 0))) ;


extern int vsscanf (const char *__restrict __s,
      const char *__restrict __format, __gnuc_va_list __arg)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__
((__format__ (__scanf__, 2, 0)));




extern int vfscanf (FILE *__restrict __s, const char *__restrict
__format, __gnuc_va_list __arg) __asm__ ("" "__isoc99_vfscanf")



     __attribute__ ((__format__ (__scanf__, 2, 0))) ;
extern int vscanf (const char *__restrict __format, __gnuc_va_list
__arg) __asm__ ("" "__isoc99_vscanf")

     __attribute__ ((__format__ (__scanf__, 1, 0))) ;
extern int vsscanf (const char *__restrict __s, const char *__restrict
__format, __gnuc_va_list __arg) __asm__ ("" "__isoc99_vsscanf")
__attribute__ ((__nothrow__ , __leaf__))
```

```
     __attribute__ ((__format__ (__scanf__, 2, 0)));
# 513 "/usr/include/stdio.h" 3 4
extern int fgetc (FILE *__stream);
extern int getc (FILE *__stream);




extern int getchar (void);




extern int getc_unlocked (FILE *__stream);
extern int getchar_unlocked (void);
# 538 "/usr/include/stdio.h" 3 4
extern int fgetc_unlocked (FILE *__stream);
# 549 "/usr/include/stdio.h" 3 4
extern int fputc (int __c, FILE *__stream);
extern int putc (int __c, FILE *__stream);




extern int putchar (int __c);
# 565 "/usr/include/stdio.h" 3 4
extern int fputc_unlocked (int __c, FILE *__stream);




extern int putc_unlocked (int __c, FILE *__stream);
extern int putchar_unlocked (int __c);




extern int getw (FILE *__stream);
```

```
extern int putw (int __w, FILE *__stream);
```

```
extern char *fgets (char *__restrict __s, int __n, FILE *__restrict
__stream)
     __attribute__ ((__access__ (__write_only__, 1, 2)));
# 632 "/usr/include/stdio.h" 3 4
extern __ssize_t __getdelim (char **__restrict __lineptr,
                             size_t *__restrict __n, int __delimiter,
                             FILE *__restrict __stream) ;
extern __ssize_t getdelim (char **__restrict __lineptr,
                           size_t *__restrict __n, int __delimiter,
                           FILE *__restrict __stream) ;
```

```
extern __ssize_t getline (char **__restrict __lineptr,
                          size_t *__restrict __n,
                          FILE *__restrict __stream) ;
```

```
extern int fputs (const char *__restrict __s, FILE *__restrict
__stream);
```

```
extern int puts (const char *__s);
```

```
extern int ungetc (int __c, FILE *__stream);




extern size_t fread (void *__restrict __ptr, size_t __size,
       size_t __n, FILE *__restrict __stream) ;



extern size_t fwrite (const void *__restrict __ptr, size_t __size,
        size_t __n, FILE *__restrict __s);
# 702 "/usr/include/stdio.h" 3 4
extern size_t fread_unlocked (void *__restrict __ptr, size_t __size,
         size_t __n, FILE *__restrict __stream) ;
extern size_t fwrite_unlocked (const void *__restrict __ptr, size_t
__size,
          size_t __n, FILE *__restrict __stream);




extern int fseek (FILE *__stream, long int __off, int __whence);



extern long int ftell (FILE *__stream) ;



extern void rewind (FILE *__stream);
# 736 "/usr/include/stdio.h" 3 4
extern int fseeko (FILE *__stream, __off_t __off, int __whence);



extern __off_t ftello (FILE *__stream) ;
```

```
# 760 "/usr/include/stdio.h" 3 4
extern int fgetpos (FILE *__restrict __stream, fpos_t *__restrict
__pos);




extern int fsetpos (FILE *__stream, const fpos_t *__pos);
# 786 "/usr/include/stdio.h" 3 4
extern void clearerr (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__));

extern int feof (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__)) ;

extern int ferror (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__)) ;



extern void clearerr_unlocked (FILE *__stream) __attribute__
((__nothrow__ , __leaf__));
extern int feof_unlocked (FILE *__stream) __attribute__ ((__nothrow__
, __leaf__)) ;
extern int ferror_unlocked (FILE *__stream) __attribute__
((__nothrow__ , __leaf__)) ;









extern void perror (const char *__s);




extern int fileno (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__)) ;




extern int fileno_unlocked (FILE *__stream) __attribute__
((__nothrow__ , __leaf__)) ;
# 823 "/usr/include/stdio.h" 3 4
extern int pclose (FILE *__stream);
```

```
extern FILE *popen (const char *__command, const char *__modes)
   __attribute__ ((__malloc__)) __attribute__ ((__malloc__ (pclose,
1))) ;




extern char *ctermid (char *__s) __attribute__ ((__nothrow__ ,
__leaf__))
   __attribute__ ((__access__ (__write_only__, 1)));
# 867 "/usr/include/stdio.h" 3 4
extern void flockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__));



extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__)) ;


extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
__leaf__));
# 885 "/usr/include/stdio.h" 3 4
extern int __uflow (FILE *);
extern int __overflow (FILE *, int);
# 902 "/usr/include/stdio.h" 3 4

# 4 "macro1.c" 2


# 5 "macro1.c"
int main(int argc, char const *argv[])
{
    printf("this is the definition of foo");
    return 0;
}
```

The function still showed up in the output, but now in the print statement FOO was replaced by the string it contained. Also the output is exponentially longer, it looks like it printed out everything within the library stdio.h.

4.

NULL is defined as:

```
#define NULL ((void *)0)
```

NULL is (void *)0 which can be converted / compared to any other type.

Nullptr can only be compared to a pointer type.

Commented [GU1]: not quite sure where to find the c++ nullptr source code

Commented [cs2R1]: it's nullptr defination in c++ is nullptr_t which is some sort of typedef thing which means dividing deeper into that.