ECE 8400 / ENGI 9875 Lab 1: C programming

Tousif Habib, Mohammed Shakeel

TOTAL POINTS

55 / 57

QUESTION 1

Pre-lab: manual pages 5 pts

1.1 man 1 man 1 / 1

✓ **- 0 pts** *Correct*

- 1 pts Not answered

1.2 Manual page sections 1 / 1

✓ **- 0 pts** *Correct*

- 1 pts Not answered

- 0.5 pts Partially answered

1.3 write(1) vs fwrite(3) 1 / 2

- 0 pts Correct

- 1 pts Not answered

✓ **- 1 pts** *Partially true*

- 1.5 pts Poor reasoning

write(1) is a user command to communicate with users, while fwrite(3) is a library function

1.4 Return type of malloc(3) 1 / 1

✓ **- 0 pts** *Correct*

- 1 pts Not answered

- 1 pts Wrong answer

QUESTION 2

2 Pre-lab: C++ functions 2 / 2

✓ **- 0 pts** *Correct*

- 0.5 pts Partially true with minor mistakes

- 2 pts Not answered

QUESTION 3

Symbols and polymorphism 11 pts

3.1 Symbols in cpp-math.o 3 / 3

✓ **- 0 pts** *Correct*

- 1 pts Missing the description

- 3 pts Not answered

3.2 Symbols with extern "C" 2 / 2

✓ **- 0 pts** *Correct*

- 1 pts Missing the description

- 0.5 pts Incomplete evidence of the work

- 2 pts Not answered

3.3 Differences with c-math.c 3 / 3

✓ **- 0 pts** *Correct*

- 1 pts True but the reasoning is not precise

- 1 pts Missing the expected evidence of the work

- 1.5 pts Poor reasoning

- 2 pts Missing the description

- 3 pts Not answered

3.4 objdump diffs 2 / 3

- 0 pts Correct

- **2 pts** Missing the description
- **1.5 pts** Not a precise explanation
- ✓ **- 1 pts** *True but not complete*
- **3 pts** Not answered

QUESTION 4

C-style strings 12 pts

4.1 fwrite(3) and string literal address **4 / 4**

- ✓ **- 0 pts** *Correct*
- **2 pts** Missing the evidence of the work
- **1.5 pts** Missing the complete debugging work
- **1.5 pts** Missing the code
- **4 pts** Not answered

4.2 Initial array **4 / 4**

- ✓ **- 0 pts** *Correct*
- **1 pts** Missing the debugging work
- **1 pts** Missing the code
- **4 pts** Not answered
- **2 pts** Missing the evidence of the work
- **4 pts** Not answered or totally wrong

4.3 strcpy(3) **4 / 4**

- ✓ **- 0 pts** *Correct*
- **2 pts** Missing the evidence of the work and debug
- **1 pts** Missing the description
- **1 pts** Incomplete reasoning
- **1 pts** Missing the code
- **4 pts** Not answered

QUESTION 5

I/O with libc 8 pts

5.1 Print an int **1 / 1**

- ✓ **- 0 pts** *Correct*
- **0.5 pts** Missing the code
- **0.5 pts** Incomplete code
- **1 pts** Not answered

5.2 Print more types **2 / 2**

- ✓ **- 0 pts** *Correct*
- **1 pts** Missing the code
- **1 pts** Incomplete code
- **2 pts** Not answered

5.3 sprintf(3) **5 / 5**

- ✓ **- 0 pts** *Correct*
- **2 pts** Setting the sprintf argument: sprintf is expected to use the varargs feature
- **3 pts** Missing the code
- **5 pts** Not answered

QUESTION 6

Memory management 11 pts

6.1 malloc(3) one integer **2 / 2**

- ✓ **- 0 pts** *Correct*
- **1 pts** Missing the code

6.2 -fsanitize=address **2 / 2**

- ✓ **- 0 pts** *Correct*
- **1 pts** Missing the evidence of the work

6.3 malloc(3) vs calloc(3) **3 / 3**

- ✓ **- 0 pts** *Correct*
- **1 pts** Wrong reasoning

- **2 pts** Missing the evidence of the work

6.4 Allocate, deallocate **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing the evidence of the work

6.5 free(3) on already-allocated array **1 / 1**

✓ **- 0 pts** *Correct*

- **0.5 pts** Wrong reasoning
- **0.5 pts** Missing the evidence of the work or code

6.6 free(3) on NULL **1 / 1**

✓ **- 0 pts** *Correct*

- **0.5 pts** Wrong answer
- **0.5 pts** Missing the evidence of the work

QUESTION 7

Macros 8 pts

7.1 #define FOO **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing the code
- **1 pts** Missing the evidence of the work and analysis
- **2 pts** Not answered

7.2 Use FOO **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing the code
- **1 pts** Missing the evidence of the work and analysis
- **2 pts** Not answered

7.3 #include <stdio.h> **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing the code
- **1 pts** Missing the evidence of the work and analysis
- **2 pts** Wrong answer: missing the main point
- **2 pts** Not answered

7.4 NULL **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing the code
- **1 pts** Missing the evidence of the work and analysis
- **1.5 pts** Poor description, missing the main points
- **0.5 pts** True, but missing minor points such as the difference in the definition between C & C++
- **2 pts** Not answered

gradescope

1.1 man 1 man **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts Not answered

gradescope

1.2 Manual page sections **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts Not answered

- 0.5 pts Partially answered

gradescope

1.3 write(1) vs fwrite(3) **1 / 2**

- **0 pts** Correct
- **1 pts** Not answered
- ✓ **- 1 pts** *Partially true*
- **1.5 pts** Poor reasoning

write(1) is a user command to communicate with users, while fwrite(3) is a library function

gradescope

1.4 Return type of malloc(3) **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts Not answered

- 1 pts Wrong answer

gradescope

2 Pre-lab: C++ functions **2 / 2**

✓ **- 0 pts** *Correct*

- 0.5 pts Partially true with minor mistakes

- 2 pts Not answered

gradescope

3.1 Symbols in cpp-math.o **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts Missing the description

- 3 pts Not answered

gradescope

3.2 Symbols with extern "C" **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the description

- 0.5 pts Incomplete evidence of the work

- 2 pts Not answered

gradescope

3.3 Differences with c-math.c **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts True but the reasoning is not precise

- 1 pts Missing the expected evidence of the work

- 1.5 pts Poor reasoning

- 2 pts Missing the description

- 3 pts Not answered

gradescope

3.4 objdump diffs **2 / 3**

- **0 pts** Correct
- **2 pts** Missing the description
- **1.5 pts** Not a precise explanation
- ✓ **- 1 pts** *True but not complete*
- **3 pts** Not answered

gradescope

4.1 fwrite(3) and string literal address **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts Missing the evidence of the work

- 1.5 pts Missing the complete debugging work

- 1.5 pts Missing the code

- 4 pts Not answered

gradescope

4.2 Initial array **4 / 4**

✓ **- 0 pts** *Correct*

- 1 pts Missing the debugging work

- 1 pts Missing the code

- 4 pts Not answered

- 2 pts Missing the evidence of the work

- 4 pts Not answered or totally wrong

gradescope

4.3 strcpy(3) **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts Missing the evidence of the work and debug

- 1 pts Missing the description

- 1 pts Incomplete reasoning

- 1 pts Missing the code

- 4 pts Not answered

gradescope

5.1 Print an int **1 / 1**

✓ **- 0 pts** *Correct*

- 0.5 pts Missing the code

- 0.5 pts Incomplete code

- 1 pts Not answered

gradescope

5.2 Print more types **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

- 1 pts Incomplete code

- 2 pts Not answered

gradescope

5.3 sprintf(3) **5 / 5**

✓ **- 0 pts** *Correct*

- 2 pts Setting the sprintf argument: sprintf is expected to use the varargs feature

- 3 pts Missing the code

- 5 pts Not answered

gradescope

6.1 malloc(3) one integer **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

gradescope

6.2 -fsanitize=address **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the evidence of the work

gradescope

6.3 malloc(3) vs calloc(3) **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts Wrong reasoning

- 2 pts Missing the evidence of the work

gradescope

6.4 Allocate, deallocate **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the evidence of the work

gradescope

6.5 free(3) on already-allocated array **1 / 1**

✓ **- 0 pts** *Correct*

- 0.5 pts Wrong reasoning

- 0.5 pts Missing the evidence of the work or code

gradescope

6.6 free(3) on NULL **1 / 1**

✓ **- 0 pts** *Correct*

- 0.5 pts Wrong answer

- 0.5 pts Missing the evidence of the work

gradescope

7.1 #define FOO **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

- 1 pts Missing the evidence of the work and analysis

- 2 pts Not answered

gradescope

7.2 Use FOO **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

- 1 pts Missing the evidence of the work and analysis

- 2 pts Not answered

gradescope

7.3 #include <stdio.h> **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

- 1 pts Missing the evidence of the work and analysis

- 2 pts Wrong answer: missing the main point

- 2 pts Not answered

gradescope

7.4 NULL **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Missing the code

- 1 pts Missing the evidence of the work and analysis

- 1.5 pts Poor description, missing the main points

- 0.5 pts True, but missing minor points such as the difference in the definition between C & C++

- 2 pts Not answered

gradescope

LAB 0

01/16/2023
ECE-8400

TOUSIF HABIB -201726429
MOHAMMED SHAKEEL - 201831104

Prelab Questions

1. The purpose of the man 1 man command is to find and display the documentation for the man command itself.

2. The difference is that section 1 describes user commands while section 3 describes library functions.

3. write(1) writes to a file descriptor. The purpose of write() is to write up to count byte(1) from the buffer pointed to the file that is referred to the by fd. If it is successful then the number of bytes that is written is returned. If an error occurs, -1 is returned.

fwrite(3) is a binary stream input/output. This function writes nmeb elements of data, where the number of bytes each character represents can be set which is obtained from the location given by ptr. On success, fwrite() returns the number of items written and this number is equivalent to number of bytes that are fed in when the size is 1. Should an error occur, the value is set to short item count or zero.

4. eturns a void pointer to the allocated memory, which is suitably aligned for any built-in type. On error, these functions return NULL.

2.1

```
#ifndef ADD_H
#define ADD_H

#include <iostream>

int add(int, int);
float add(float, float);

#endif

#include "add.h"

int add(int a, int b)
{
          return int(a + b);
}

float add(float a, float b)
{
          return float(a + b);
}
```

Symbols and Polymorphism

1. Input:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ nm cpp-math.o | c++filt
```

Output:

```
0000000000000000 T add(int, int)
000000000000001e T add(float, float)
```

The demangled output output shows that there are two function signatures with the same name however one accepts int types as parameters and the other accepts float types as parameters.

2. Input:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ nm cpp-math.o | c++filt
```

Output:

```
0000000000000000 T add
000000000000001e T add(float, float)
```

It seems that the argument for the int type has disappeared

3. Input:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ gcc -c c-math.c
```

Output:

```
c-math.c:8:7: error: conflicting types for ‘add’
          8 | float add(float varFloat1, float varFloat2){
            |         ^~~
c-math.c:3:5: note: previous definition of ‘add’ was here
          3 | int add(int varInt1, int varInt2){
            |         ^~~
```

After putting the same code into and .c file it is observed that the add function no longer compiles due to c not supporting operator overloading.

4.

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ objdump -dS c-math.o | pygmentize -l c-objdump
c-math.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <add>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   89 7d ec                mov    %edi,-0x14(%rbp)
   b:   89 75 e8                mov    %esi,-0x18(%rbp)
   e:   8b 55 ec                mov    -0x14(%rbp),%edx
  11:   8b 45 e8                mov    -0x18(%rbp),%eax
  14:   01 d0                   add    %edx,%eax
  16:   89 45 fc                mov    %eax,-0x4(%rbp)
  19:   8b 45 fc                mov    -0x4(%rbp),%eax
  1c:   5d                      pop    %rbp
  1d:   c3                      retq
```

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ objdump -dS cpp-math.o | pygmentize -l c-objdump
cpp-math.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <add>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   89 7d ec                mov    %edi,-0x14(%rbp)
   b:   89 75 e8                mov    %esi,-0x18(%rbp)
   e:   8b 55 ec                mov    -0x14(%rbp),%edx
  11:   8b 45 e8                mov    -0x18(%rbp),%eax
  14:   01 d0                   add    %edx,%eax
  16:   89 45 fc                mov    %eax,-0x4(%rbp)
  19:   8b 45 fc                mov    -0x4(%rbp),%eax
  1c:   5d                      pop    %rbp
  1d:   c3                      retq

000000000000001e <_Z3addff>:
  1e:   f3 0f 1e fa             endbr64
  22:   55                      push   %rbp
  23:   48 89 e5                mov    %rsp,%rbp
  26:   f3 0f 11 45 ec          movss  %xmm0,-0x14(%rbp)
  2b:   f3 0f 11 4d e8          movss  %xmm1,-0x18(%rbp)
  30:   f3 0f 10 45 ec          movss  -0x14(%rbp),%xmm0
  35:   f3 0f 58 45 e8          addss  -0x18(%rbp),%xmm0
  3a:   f3 0f 11 45 fc          movss  %xmm0,-0x4(%rbp)
  3f:   f3 0f 10 45 fc          movss  -0x4(%rbp),%xmm0
  44:   5d                      pop    %rbp
  45:   c3                      retq
```

It seems that the C++ implementation has an extra function signature that is used to handle floats and as such uses movss and addss vs mov and add because those are special x86 assembly commands for floating point calculations.

C-Style Strings

1. Code:

```
#include <stdio.h>
#include <string.h>

int main(){
   char a[] = "Operating systems rule";
   fwrite(a,1,sizeof(a), stdout);
   return 0;
};
```

Input/Output:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ gdb Box
(gdb) break c-style-string.c:4
Breakpoint 1 at 0x1184: file c-style-string.c, line 4.
(gdb) info break
Num       Type           Disp Enb Address            What
1         breakpoint     keep y   0x0000000000001184 in main at c-style-string.c:4
(gdb) run
Starting program: /users/labnet4/st6/thabib/OS Labs/lab1/Box
Breakpoint 1, main () at c-style-string.c:4
4         char a[] = "Operating systems rule";
(gdb) n
5         fwrite(a,1,sizeof(a), stdout);
(gdb) print a
$1 = "Operating systems rule"
(gdb) print &a
$2 = (char (*)[23]) 0x7fffffffe0d0
(gdb)
```

The address is 0x7fffffffe0d0

2. Input/Output:

```
Starting program: /users/labnet4/st6/thabib/OS Labs/lab1/Dox
Breakpoint 2, main () at c-style-string.c:5
5         int len = strlen(initials);
(gdb) print initials
$4 = "MSS"
(gdb) print &initials
$5 = (char (*)[4]) 0x7fffffffe0e4
(gdb) x/8x &initials
0x7fffffffe0e4: 0x0053534d   0x63a78e00         0x6b71b0e1         0x00000000
0x7fffffffe0f4: 0x00000000   0xf7dbc083         0x00007fff         0xf7ffc620
```

Little-endian so read from right to left
4d = M
53 = S
53 = S
00 = NULL

3. Input/Output:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ gdb Aox
(gdb) break c-style-string.c:4
Breakpoint 1 at 0x11a4: file c-style-string.c, line 4.
(gdb) info break
Num       Type           Disp Enb Address            What
1         breakpoint     keep y   0x00000000000011a4 in main at c-style-string.c:4
(gdb) run
Starting program: /users/labnet4/st6/thabib/OS Labs/lab1/Aox

Breakpoint 1, main () at c-style-string.c:4
4         char initials[] = "MSS";
(gdb) n
5         char name[] = "Shakeel";
(gdb) print initials
$1 = "MSS"
(gdb) n
6         strcpy(initials,name);
(gdb) print initials
$2 = "MSS"
(gdb) n
7         int len = strlen(initials);
(gdb) print initials
$3 = "Shak"
(gdb) print &initials
$4 = (char (*)[4]) 0x7fffffffe0dc
(gdb) x/8x 0x7fffffffe0dc
0x7fffffffe0dc: 0x6b616853   0x006c6565         0x006c6565         0xdc06de00
0x7fffffffe0ec: 0x96a07882   0x00000000         0x00000000         0xf7dbc083
```

From observation of debugger output it seems that only the first four bytes are copied. As such there is reason to believe that buffer overflow occurs.

I/O with libc

1. Code:

```
#include <stdio.h>
#include <string.h>
int main(){
   printf("My fav. integer is: %d", 12);
};
```

Input:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ ./first
```

Output:

```
My favourite integer is: 12
```

2. Code:

```
#include <stdio.h>
#include <string.h>
int main(){
   int *hello = 24;
   printf("My favourite integer is: %d \n", 12);
   printf("My favourite integer is: %f \n", 12.23);
   printf("My favourite integer is: %04x \n", 65536);
   printf("My favourite integer is: %p \n", hello);
};
```

Input:

```
thabib@slbncsf2112pc06:~/OS Labs/lab1$ ./Hello
```

Output:

```
My favourite integer is: 12
My favourite integer is: 12.230000
My favourite integer is: 10000
My favourite integer is: 0x18
```

3. Code:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(){
   char str[128];
   char name[] = "Mohammed Shakeel";
   int len = strlen(name);
   char *ptr = name;
   sprintf(str,"my name is %s, which is %d characters long, and it is located at address %p", name, len, ptr);

   fwrite(ptr, 1, sizeof(str), stdout);

   write(str, sizeof(str), STDOUT_FILENO);

};
```

Input:

```
tousifhabib@LAPTOP-QTNGMCC0:~/ECE$ gcc -g sample.c -o sample
tousifhabib@LAPTOP-QTNGMCC0:~/ECE$ ./sample
```

Ouput:

```
my name is Mohammed Shakeel, which is 16 characters long, and it is located at address 0x7fffef
```

Memory Management

1. Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
   int* intPtr;
   intPtr = (int *)malloc(sizeof(int));
   printf("Address of the int variable is: %p\n", intPtr);
   return 0;
}
```

Input:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -g  test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
```

Output:

```
Address of the int variable is: 0x56553f2b92a0
```

2. Output:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -fsanitize=address -g test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
Address of the int variable is: 0x602000000010

=================================================================
==46657==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 4 byte(s) in 1 object(s) allocated from:
    #0 0x7f8a90f72808 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x55deddfa21fe in main /users/labnet/st3/msshakeel/operatingLabs/lab1/test.c:7
    #2 0x7f8a90c97082 in __libc_start_main ../csu/libc-start.c:308

SUMMARY: AddressSanitizer: 4 byte(s) leaked in 1 allocation(s).
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$
```

Since there is no freeing of the memory it results in memory leak detected by the sanitizer. We also observe that the address space has changed.

3. Code:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
   int* intArrayMal = (int*) malloc(1000* sizeof(int));
   int* intArrayCal = (int*) calloc(1000,sizeof(int));
   return 0;
};
```

Output:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -g  test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gdb output
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from output...
(gdb) break main
Breakpoint 1 at 0x1169: file test.c, line 5.
(gdb) run
Starting program: /users/labnet/st3/msshakeel/operatingLabs/lab1/output

Breakpoint 1, main () at test.c:5
5       int main(){
(gdb) n
6           int* intArrayMal = (int*) malloc(1000* sizeof(int));
(gdb) n
7           int* intArrayCal = (int*) calloc(1000,sizeof(int));
(gdb) print intArrayCal
$1 = (int *) 0x0
(gdb) print intArrayMal
$2 = (int *) 0x5555555592a0
(gdb) print *intArrayMal
$3 = 0
(gdb) print *intArrayCal
Cannot access memory at address 0x0
(gdb)
```

4. Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  char* array = (char*) malloc(100 * sizeof(char));
  printf("value of array pointer: %p\n", array);

  free(array);
  printf("Value of array ptr after freeing: %p\n", array);
  return 0;
};
```

Input:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -g  test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
```

Output:

```
value of array pointer: 0x55e9bbe282a0
Value of array ptr after freeing: 0x55e9bbe282a0
```

5. Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  char* array = (char*) malloc(100 * sizeof(char));
  printf("value of array pointer: %p\n", array);

  free(array);
  printf("Value of array ptr after freeing: %p\n", array);
  free(array);
  printf("Value of array ptr after freeing second time: %p\n", array);
  return 0;
};
```

Input:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -g  test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
```

Output:

```
value of array pointer: 0x565424edf2a0
Value of array ptr after freeing: 0x565424edf2a0
free(): double free detected in tcache 2
Aborted (core dumped)
```

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

6. Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  char* array = (char*) malloc(100 * sizeof(char));
  printf("value of array pointer: %p\n", array);
  array = NULL;
  printf("value of array pointer when initialized to null: %p\n", array);
  free(array);
   printf("value of null array pointer after freeing: %p\n", array);
  return 0;
};
```

Input:

```
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ gcc -g  test.c -o output
msshakeel@slbncsf2112pc06:~/operatingLabs/lab1$ ./output
```

Output:

```
value of array pointer: 0x562a3978c2a0
value of array pointer when initialized to null: (nil)
value of null array pointer after freeing: (nil)
```

It is observed that calling free(3) on a NULL pointer is allowed and there is no change in address space.

Macros

1. C file:

```
#define FOO "this is the definition of foo"
```

Input:

```
tousifhabib@LAPTOP-QTNGMCC0:~/ECE$ cc -E sample.c
```

Output:

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

The more verbose c file does not contain the string FOO.

2. C file:

```
#define FOO "this is the definition of foo"

char FOOuser(char FOO){
    return(FOO)
}
```

Input:

```
tousifhabib@LAPTOP-QTNGMCC0:~/ECE$ cc -E sample.c
```

Output:

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

char FOOuser(char "this is the definition of foo"){
    return("this is the definition of foo")
}
```

What is different here is that when FOO is passed into the function, in the output the alias for FOO is displayed as the defined value “this is the definition as foo”, FOO itself does not appear.

3. C file:

```
#include <stdio.h>
#define FOO "this is the definition of foo"

char FOOuser(char FOO){
    return(FOO)
}

int main()
{
    printf(FOO);
    return 0;
}
```

Input:

```
tousifhabib@LAPTOP-QTNGMCC0:~/ECE$ cc -E sample.c
```

Ouptut:

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

# 2 "sample.c" 2

# 4 "sample.c"
char FOOuser(char "this is the definition of foo"){
    return("this is the definition of foo")
}

int main()
{
    printf("this is the definition of foo");
    return 0;
}
```

In the preprocessed output the source code for stdio.h is output that is relevant for printf.

4. In the location: `/usr/src/linux-headers-XXXX/include/linux/stddef.h`

Null is defined as:

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

However in C++ NULL implicitly means 0 and the nullptr is used because unlike C, there is no implied conversion from void * to other pointer types. Nullptr is convertible to other pointer types but not int, this is useful for overload resolution.