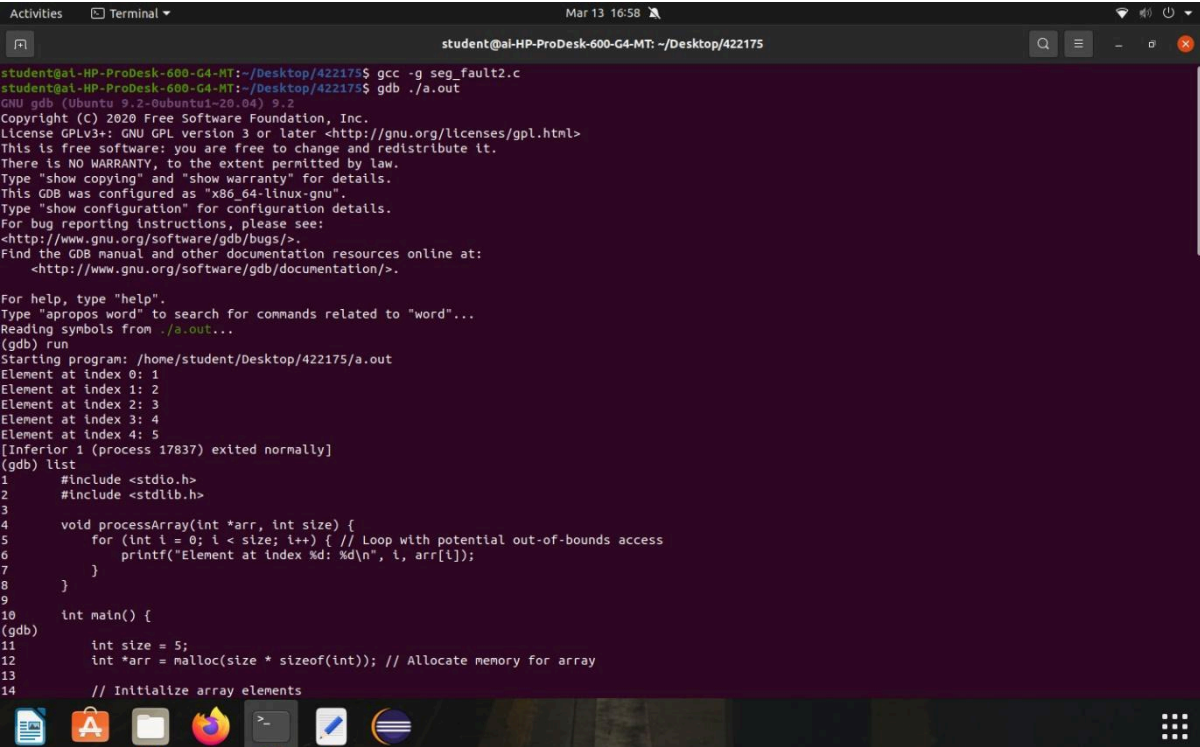


k.vamshi

Roll No:422175

Unixlab Assignment

//seg_fault2.c : OUTPUT->



```
student@ai-HP-ProDesk-600-G4-MT: ~/Desktop/422175$ gcc -g seg_fault2.c
student@ai-HP-ProDesk-600-G4-MT: ~/Desktop/422175$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 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 ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422175/a.out
Element at index 0: 1
Element at index 1: 2
Element at index 2: 3
Element at index 3: 4
Element at index 4: 5
[Inferior 1 (process 17837) exited normally]
(gdb) list
1      #include <stdio.h>
2      #include <stdlib.h>
3
4      void processArray(int *arr, int size) {
5          for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
6              printf("Element at index %d: %d\n", i, arr[i]);
7          }
8      }
9
10     int main() {
(gdb)
11         int size = 5;
12         int *arr = malloc(size * sizeof(int)); // Allocate memory for array
13
14         // Initialize array elements
```

```
Activities Terminal Mar 13 16:58 student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175
13 // Initialize array elements
14 for (int i = 0; i < size; i++) {
15     arr[i] = i + 1;
16 }
17
18 processArray(arr, size); // Pass the array to processArray
19
20 (gdb)
21 free(arr); // Free allocated memory
22 return 0;
23 }
24 (gdb)
Line number 24 out of range; seg_fault2.c has 23 lines.
(gdb) b
No default breakpoint address now.
(gdb) break processArray
Breakpoint 1 at 0x55555555189: file seg_fault2.c, line 4.
(gdb) break 17
Breakpoint 2 at 0x5555555523a: file seg_fault2.c, line 19.
(gdb) run
Starting program: /home/student/Desktop/422175/a.out
Breakpoint 2, main () at seg_fault2.c:19
19 processArray(arr, size); // Pass the array to processArray
(gdb) next
Breakpoint 1, processArray (arr=0x7ffff7e51ba2 <malloc_hook_int+146>, size=0) at seg_fault2.c:4
4 void processArray(int *arr, int size) {
(gdb) next
5 for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
(gdb) next
6     printf("Element at index %d: %d\n", i, arr[i]);
(gdb) next
Element at index 0: 1
5 for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
(gdb) print i
$1 = 0
(gdb) print i
$2 = 0
(gdb) next
6     printf("Element at index %d: %d\n", i, arr[i]);
(gdb) next
```

```
Activities Terminal Mar 13 16:58 student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175
5 for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
(gdb) next
6     printf("Element at index %d: %d\n", i, arr[i]);
(gdb) next
Element at index 0: 1
5 for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
(gdb) print i
$1 = 0
(gdb) print i
$2 = 0
(gdb) next
6     printf("Element at index %d: %d\n", i, arr[i]);
(gdb) print i
$3 = 1
(gdb) next
Element at index 1: 2
5 for (int i = 0; i < size; i++) { // Loop with potential out-of-bounds access
(gdb) print i
$4 = 1
(gdb) next
6     printf("Element at index %d: %d\n", i, arr[i]);
(gdb) continue
Continuing.
Element at index 2: 3
Element at index 3: 4
Element at index 4: 5
[Inferior 1 (process 17900) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
0x0000555555551e1: <+0>: endbr64
0x0000555555551e5: <+4>: push %rbp
0x0000555555551e6: <+5>: mov %rsp,%rbp
0x0000555555551e9: <+8>: sub $0x10,%rsp
0x0000555555551ed: <+12>: movl $0x5,-0xc(%rbp)
0x0000555555551f4: <+19>: mov -0xc(%rbp),%eax
0x0000555555551f7: <+22>: cltq
0x0000555555551f9: <+24>: shl $0x2,%rax
0x0000555555551fd: <+28>: mov %rax,%rdi
0x000055555555200: <+31>: callq 0x55555555090 <malloc@plt>
0x000055555555205: <+36>: mov %rax,-0x8(%rbp)
0x000055555555209: <+40>: movl $0x0,-0x10(%rbp)
0x000055555555210: <+47>: jmp 0x55555555232 <main+81>
```

```
Activities Terminal Mar 13 16:58
student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175

Element at index 4: 5
[Inferior 1 (process 17900) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
0x0000555555551e1 <+0>: endbr64
0x0000555555551e5 <+4>: push %rbp
0x0000555555551e6 <+5>: mov %rsp,%rbp
0x0000555555551e9 <+8>: sub $0x10,%rsp
0x0000555555551ed <+12>: movl $0x5,-0xc(%rbp)
0x0000555555551f4 <+19>: mov -0xc(%rbp),%eax
0x0000555555551f7 <+22>: cltq
0x0000555555551f9 <+24>: shl $0x2,%rax
0x0000555555551fd <+28>: mov %rax,%rdi
0x000055555555200 <+31>: callq 0x555555555090 <malloc@plt>
0x000055555555205 <+36>: mov %rax,-0x8(%rbp)
0x000055555555209 <+40>: movl $0x0,-0x10(%rbp)
0x000055555555210 <+47>: jmp 0x555555555232 <main+81>
0x000055555555212 <+49>: mov -0x10(%rbp),%eax
0x000055555555215 <+52>: cltq
0x000055555555217 <+54>: lea 0x0(,%rax,4),%rdx
0x00005555555521f <+62>: mov -0x8(%rbp),%rax
0x000055555555223 <+66>: add %rdx,%rax
0x000055555555226 <+69>: mov -0x10(%rbp),%edx
0x000055555555229 <+72>: add $0x1,%edx
0x00005555555522c <+75>: mov %edx,%rax
0x00005555555522e <+77>: addl $0x1,-0x10(%rbp)
0x000055555555232 <+81>: mov -0x10(%rbp),%eax
0x000055555555235 <+84>: cmp -0xc(%rbp),%eax
0x000055555555238 <+87>: jle 0x555555555212 <main+49>
0x00005555555523a <+89>: mov -0xc(%rbp),%edx
0x00005555555523d <+92>: mov -0x8(%rbp),%rax
0x000055555555241 <+96>: mov %edx,%esi
0x000055555555243 <+98>: mov %rax,%rdi
0x000055555555246 <+101>: callq 0x555555555189 <processArray>
0x00005555555524b <+106>: mov -0x8(%rbp),%rax
0x00005555555524f <+110>: mov %rax,%rdi
0x000055555555252 <+113>: callq 0x555555555070 <free@plt>
0x000055555555257 <+118>: mov $0x0,%eax
0x00005555555525c <+123>: leaveq
0x00005555555525d <+124>: retq
End of assembler dump.
(gdb) 
```

//def_Null.c:output->

```
Activities Terminal Mar 13 16:40
student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175

student@al-HP-ProDesk-600-G4-MT:~/Desktop/422175$ gcc -g def_Null.c
student@al-HP-ProDesk-600-G4-MT:~/Desktop/422175$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1-20.04) 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 ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422175/a.out

Program received signal SIGSEGV, Segmentation fault.
0x00005555555513d in main () at def_Null.c:5
5      *ptr = 10; // Dereferencing NULL pointer
(gdb) list
1      #include <stdio.h>
2
3      int main() {
4          int *ptr = NULL;
5          *ptr = 10; // Dereferencing NULL pointer
6          return 0;
7      }
(gdb)
Line number 8 out of range; def_Null.c has 7 lines.
(gdb) break main
Breakpoint 1 at 0x55555555129: file def_Null.c, line 3.
(gdb) break 6
Breakpoint 2 at 0x55555555143: file def_Null.c, line 6.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422175/a.out
```

```
Activities Terminal Mar 13 16:41 student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175

(gdb) break 6
Breakpoint 2 at 0x55555555143: file def_Null.c, line 6.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422175/a.out

Breakpoint 1, main () at def_Null.c:3
3   int main() {
(gdb) next
4       int *ptr = NULL;
(gdb) print ptr
$1 = (int *) 0x0
(gdb) print *ptr
Cannot access memory at address 0x0
(gdb) next
5       *ptr = 10; // Dereferencing NULL pointer
(gdb) print &ptr
$2 = (int **) 0x7fffffffde68
(gdb) next

Program received signal SIGSEGV, Segmentation fault.
0x00005555555513d in main () at def_Null.c:5
5       *ptr = 10; // Dereferencing NULL pointer
(gdb) continue
Continuing.

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) disassemble main
Dump of assembler code for function main:
0x000055555555129 <+0>:    endbr64
0x00005555555512d <+4>:    push    %rbp
0x00005555555512e <+5>:    mov     %rsp,%rbp
0x000055555555131 <+8>:    movq    $0x0,-0x8(%rbp)
0x000055555555139 <+16>:   mov     -0x8(%rbp),%rax
0x00005555555513d <+20>:   movl    $0xa,%eax
0x000055555555143 <+26>:   mov     $0x0,%eax
0x000055555555148 <+31>:   pop     %rbp
0x000055555555149 <+32>:   retq
End of assembler dump.
(gdb)
```

//seg_error.c output->

```
student@al-HP-ProDesk-600-G4-MT:~/Desktop/422175$ gcc -g seg_error.c
student@al-HP-ProDesk-600-G4-MT:~/Desktop/422175$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu2~20.04) 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/>.

or help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422175/a.out

Program received signal SIGSEGV, Segmentation fault.
0x0000555555551d5 in findMedian (arr=0x0, size=5) at seg_error.c:11
1       return arr[size / 2]; // Accessing middle element for odd-sized array
(gdb) list
1       return arr[size / 2]; // Accessing middle element for odd-sized array
2
3   float findMedian(int *arr, int size) {
4       if (size % 2 == 0) {
5           return (arr[size / 2] + arr[(size / 2) - 1]) / 2.0; // Accessing middle elements for even-sized array
6       } else {
7           return arr[size / 2]; // Accessing middle element for odd-sized array
8       }
9   }
10
11 int main() {
12     int *arr = NULL; // Set the array pointer to NULL
13     int size = 5; // Array size
14
15     float median = findMedian(arr, size); // Pass NULL pointer to findMedian
16     printf("Median: %.2f\n", median); // Attempt to print the median (segmentation fault expected)
17     return 0;
18 }
```



```
Activities Terminal Mar 13 16:20 student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175
20 printf("Median: %.2f\n", median); // Attempt to print the median (segmentation fault expected)
21 return 0;
22 }
(gdb)
Line number 23 out of range; seg_error.c has 22 lines.
(gdb) break 6
Breakpoint 1 at 0x55555555149: file seg_error.c, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422175/a.out

Breakpoint 1, findMedian (arr=0x55555555230 <_libc_csu_init>, size=32767)
7 at seg_error.c:7
float findMedian(int *arr, int size) {
(gdb) break main
Breakpoint 2 at 0x555555551dd: file seg_error.c, line 15.
(gdb) break findMedian
Note: breakpoint 1 also set at pc 0x55555555149.
Breakpoint 3 at 0x55555555149: file seg_error.c, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422175/a.out

Breakpoint 2, main () at seg_error.c:15
15 int main() {
(gdb) next
16 int *arr = NULL; // Set the array pointer to NULL
(gdb) print arr
$1 = (int *) 0x0
(gdb) next
17 int size = 5; // Array size
(gdb) print size
$2 = -8352
(gdb) next
19 float median = findMedian(arr, size); // Pass NULL pointer to findMedian
(gdb) print median
$3 = 4.59163468e-41
(gdb) next

Breakpoint 1, findMedian (arr=0x55555555230 <_libc_csu_init>, size=32767)
```

```
Activities Terminal Mar 13 16:21 student@al-HP-ProDesk-600-G4-MT: ~/Desktop/422175
17 int size = 5; // Array size
(gdb) print size
$2 = -8352
(gdb) next
19 float median = findMedian(arr, size); // Pass NULL pointer to findMedian
(gdb) print median
$3 = 4.59163468e-41
(gdb) next

Breakpoint 1, findMedian (arr=0x55555555230 <_libc_csu_init>, size=32767)
7 at seg_error.c:7
float findMedian(int *arr, int size) {
(gdb) continue
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x0000555555551d5 in findMedian (arr=0x0, size=5) at seg_error.c:11
11 return arr[size / 2]; // Accessing middle element for odd-sized array
(gdb) disassemble main
Dump of assembler code for function main:
0x0000555555551dd <+0>: endbr64
0x0000555555551e1 <+4>: push %rbp
0x0000555555551e2 <+5>: mov %rsp,%rbp
0x0000555555551e5 <+8>: sub $0x10,%rsp
0x0000555555551e9 <+12>: movq $0x0,-0x8(%rbp)
0x0000555555551f1 <+20>: movl $0x5,-0x10(%rbp)
0x0000555555551f8 <+27>: mov -0x10(%rbp),%edx
0x0000555555551fb <+30>: mov -0x8(%rbp),%rax
0x0000555555551ff <+34>: mov %edx,%esi
0x000055555555201 <+36>: mov %rax,%rdi
0x000055555555204 <+39>: callq 0x55555555149 <findMedian>
0x000055555555209 <+44>: movd %xmm0,%eax
0x00005555555520d <+48>: mov %eax,-0xc(%rbp)
0x000055555555210 <+51>: cvtss2sd -0xc(%rbp),%xmm0
0x000055555555215 <+56>: lea 0xdec(%rip),%rdi # 0x55555555008
0x00005555555521c <+63>: mov $0x1,%eax
0x000055555555221 <+68>: callq 0x55555555050 <printf@plt>
0x000055555555226 <+73>: mov $0x0,%eax
0x00005555555522b <+78>: leaveq
0x00005555555522c <+79>: retq
End of assembler dump.
(gdb)
```