

# Unix week-9

# GNU Debugger

Program showing NULL pointer access.

**example.c**  
~/Desktop/scripts/week1

Ln 19, Col 6

example3.c

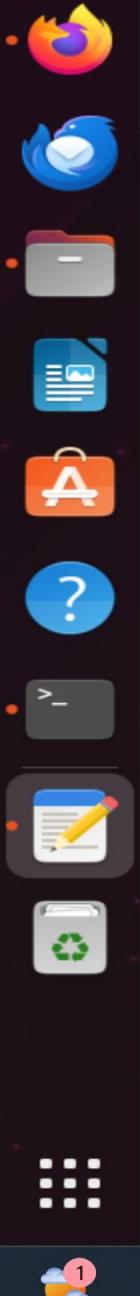
main.c

segment.

example.

### example2.

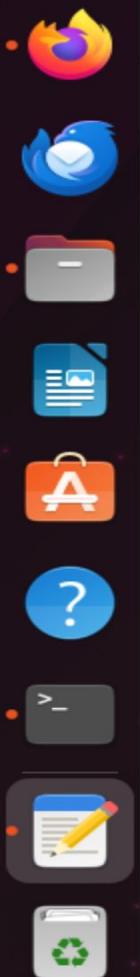
example3.c



**example.c**  
~/Desktop/scripts/week1

Ln 19, Col 6

example3.c



sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

```
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gcc -g example.c
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gdb ./a.out
GNU gdb (Ubuntu 13.1-2ubuntu2.1) 13.1
Copyright (C) 2023 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:
<https://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/sridhar/Desktop/scripts/week7/a.out

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
```



Search



Activities Terminal



```
sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7
array[8] = 16
array[9] = 18
Trying to access the array after freeing it:

Program received signal SIGSEGV, Segmentation fault.
0x00005555555526f in main () at example.c:32
32          printf("array[%d] = %d\n", i, array[i]);
(gdb) list
27          array = NULL;
28
29          // Attempt to access the array after freeing it (causing segmentation fault)
30          printf("Trying to access the array after freeing it:\n");
31          for (int i = 0; i < SIZE; i++) {
32              printf("array[%d] = %d\n", i, array[i]);
33          }
34
35      return 0;
36  }
(gdb) break 26
Breakpoint 1 at 0x5555555523b: file example.c, line 27.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sridhar/Desktop/scripts/week7/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
array[7] = 14
array[8] = 16
```



Search

ENG  
IN14:47  
13-03-202414:47  
13-03-2024

PRE



```
array[6] = 12
array[7] = 14
array[8] = 16
array[9] = 18
```

```
Breakpoint 1, main () at example.c:27
27          array = NULL;
```

```
(gdb) continue
Continuing.
```

```
Trying to access the array after freeing it:
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
0x00005555555526f in main () at example.c:32
32          printf("array[%d] = %d\n", i, array[i]);
(gdb) next
```

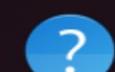
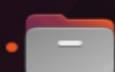
```
Program terminated with signal SIGSEGV, Segmentation fault.
```

```
The program no longer exists.
```

```
(gdb) disassemble main
```

```
Dump of assembler code for function main:
```

```
0x000055555555189 <+0>:    endbr64
0x00005555555518d <+4>:    push   %rbp
0x00005555555518e <+5>:    mov    %rsp,%rbp
0x000055555555191 <+8>:    sub    $0x20,%rsp
0x000055555555195 <+12>:   movq   $0x0,-0x8(%rbp)
0x00005555555519d <+20>:   mov    $0x28,%edi
0x0000555555551a2 <+25>:   call   0x55555555090 <malloc@plt>
0x0000555555551a7 <+30>:   mov    %rax,-0x8(%rbp)
0x0000555555551ab <+34>:   cmpq   $0x0,-0x8(%rbp)
0x0000555555551b0 <+39>:   jne    0x555555551cb <main+66>
0x0000555555551b2 <+41>:   lea    0xe4f(%rip),%rax      # 0x555555556008
0x0000555555551b9 <+48>:   mov    %rax,%rdi
0x0000555555551bc <+51>:   call   0x55555555070 <puts@plt>
0x0000555555551c1 <+56>:   mov    $0x1,%eax
0x0000555555551c6 <+61>:   imr   0x555555555200 <main+272>
```





[+] sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

```
0x000055555555189 <+0>:    endbr64
0x00005555555518d <+4>:    push   %rbp
0x00005555555518e <+5>:    mov    %rsp,%rbp
0x000055555555191 <+8>:    sub    $0x20,%rsp
0x000055555555195 <+12>:   movq   $0x0,-0x8(%rbp)
0x00005555555519d <+20>:   mov    $0x28,%edi
0x0000555555551a2 <+25>:   call   0x55555555090 <malloc@plt>
0x0000555555551a7 <+30>:   mov    %rax,-0x8(%rbp)
0x0000555555551ab <+34>:   cmpq   $0x0,-0x8(%rbp)
0x0000555555551b0 <+39>:   jne    0x555555551cb <main+66>
0x0000555555551b2 <+41>:   lea    0xe4f(%rip),%rax      # 0x555555556008
0x0000555555551b9 <+48>:   mov    %rax,%rdi
0x0000555555551bc <+51>:   call   0x55555555070 <puts@plt>
0x0000555555551c1 <+56>:   mov    $0x1,%eax
0x0000555555551c6 <+61>:   jmp    0x55555555299 <main+272>
0x0000555555551cb <+66>:   movl   $0x0,-0x14(%rbp)
0x0000555555551d2 <+73>:   jmp    0x555555551f3 <main+106>
0x0000555555551d4 <+75>:   mov    -0x14(%rbp),%eax
0x0000555555551d7 <+78>:   cltq
0x0000555555551d9 <+80>:   lea    0x0(%rax,4),%rdx
0x0000555555551e1 <+88>:   mov    -0x8(%rbp),%rax
0x0000555555551e5 <+92>:   add    %rdx,%rax
0x0000555555551e8 <+95>:   mov    -0x14(%rbp),%edx
0x0000555555551eb <+98>:   add    %edx,%edx
0x0000555555551ed <+100>:  mov    %edx,(%rax)
0x0000555555551ef <+102>:  addl   $0x1,-0x14(%rbp)
0x0000555555551f3 <+106>:  cmpl   $0x9,-0x14(%rbp)
0x0000555555551f7 <+110>:  jle    0x555555551d4 <main+75>
0x0000555555551f9 <+112>:  movl   $0x0,-0x10(%rbp)
0x000055555555200 <+119>:  jmp    0x55555555235 <main+172>
0x000055555555202 <+121>:  mov    -0x10(%rbp),%eax
0x000055555555205 <+124>:  cltq
--Type <RET> for more, q to quit, c to continue without paging--c
0x000055555555207 <+126>:  lea    0x0(%rax,4),%rdx
0x00005555555520f <+128>:  mov    -0x8(%rbp),%rax
```



Search

14:48  
13-03-2024ENG  
IN



```
sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7
0x000055555555202 <+121>: mov    -0x10(%rbp),%eax
0x000055555555205 <+124>: cltq
--Type <RET> for more, q to quit, c to continue without paging--c
0x000055555555207 <+126>: lea    0x0(%rax,4),%rdx
0x00005555555520f <+134>: mov    -0x8(%rbp),%rax
0x000055555555213 <+138>: add    %rdx,%rax
0x000055555555216 <+141>: mov    (%rax),%edx
0x000055555555218 <+143>: mov    -0x10(%rbp),%eax
0x00005555555521b <+146>: mov    %eax,%esi
0x00005555555521d <+148>: lea    0xe09(%rip),%rax      # 0x5555555602d
0x000055555555224 <+155>: mov    %rax,%rdi
0x000055555555227 <+158>: mov    $0x0,%eax
0x00005555555522c <+163>: call   0x55555555080 <printf@plt>
0x000055555555231 <+168>: addl   $0x1,-0x10(%rbp)
0x000055555555235 <+172>: cmpl   $0x9,-0x10(%rbp)
0x000055555555239 <+176>: jle    0x55555555202 <main+121>
0x00005555555523b <+178>: movq   $0x0,-0x8(%rbp)
0x000055555555243 <+186>: lea    0xdf6(%rip),%rax      # 0x55555556040
0x00005555555524a <+193>: mov    %rax,%rdi
0x00005555555524d <+196>: call   0x55555555070 <puts@plt>
0x000055555555252 <+201>: movl   $0x0,-0xc(%rbp)
0x000055555555259 <+208>: jmp    0x5555555528e <main+261>
0x00005555555525b <+210>: mov    -0xc(%rbp),%eax
0x00005555555525e <+213>: cltq
0x000055555555260 <+215>: lea    0x0(%rax,4),%rdx
0x000055555555268 <+223>: mov    -0x8(%rbp),%rax
0x00005555555526c <+227>: add    %rdx,%rax
0x00005555555526f <+230>: mov    (%rax),%edx
0x000055555555271 <+232>: mov    -0xc(%rbp),%eax
0x000055555555274 <+235>: mov    %eax,%esi
0x000055555555276 <+237>: lea    0xdb0(%rip),%rax      # 0x5555555602d
0x00005555555527d <+244>: mov    %rax,%rdi
0x000055555555280 <+247>: mov    $0x0,%eax
0x000055555555285 <+252>: call   0x55555555080 <printf@plt>
0x000055555555288 <+257>: addl   $0x1,0x4(%rbp)
```



Search

ENG  
IN14:48  
13-03-2024

PRE



sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

```
0x00005555555521b <+146>: mov    %eax,%esi
0x00005555555521d <+148>: lea    0xe0(%rip),%rax      # 0x55555555602d
0x000055555555224 <+155>: mov    %rax,%rdi
0x000055555555227 <+158>: mov    $0x0,%eax
0x00005555555522c <+163>: call   0x55555555080 <printf@plt>
0x000055555555231 <+168>: addl   $0x1,-0x10(%rbp)
0x000055555555235 <+172>: cmpl   $0x9,-0x10(%rbp)
0x000055555555239 <+176>: jle    0x55555555202 <main+121>
0x00005555555523b <+178>: movq   $0x0,-0x8(%rbp)
0x000055555555243 <+186>: lea    0xdf6(%rip),%rax      # 0x555555556040
0x00005555555524a <+193>: mov    %rax,%rdi
0x00005555555524d <+196>: call   0x55555555070 <puts@plt>
0x000055555555252 <+201>: movl   $0x0,-0xc(%rbp)
0x000055555555259 <+208>: jmp    0x5555555528e <main+261>
0x00005555555525b <+210>: mov    -0xc(%rbp),%eax
0x00005555555525e <+213>: cltq
0x000055555555260 <+215>: lea    0x0(%rax,4),%rdx
0x000055555555268 <+223>: mov    -0x8(%rbp),%rax
0x00005555555526c <+227>: add    %rdx,%rax
0x00005555555526f <+230>: mov    (%rax),%edx
0x000055555555271 <+232>: mov    -0xc(%rbp),%eax
0x000055555555274 <+235>: mov    %eax,%esi
0x000055555555276 <+237>: lea    0xdb0(%rip),%rax      # 0x55555555602d
0x00005555555527d <+244>: mov    %rax,%rdi
0x000055555555280 <+247>: mov    $0x0,%eax
0x000055555555285 <+252>: call   0x55555555080 <printf@plt>
0x00005555555528a <+257>: addl   $0x1,-0xc(%rbp)
0x00005555555528e <+261>: cmpl   $0x9,-0xc(%rbp)
0x000055555555292 <+265>: jle    0x5555555525b <main+210>
0x000055555555294 <+267>: mov    $0x0,%eax
0x000055555555299 <+272>: leave 
0x00005555555529a <+273>: ret
```

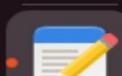
End of assembler dump.  
(gdb)



Search

ENG  
IN14:48  
13-03-2024

# Illegal Memory Access



Open

example2.c  
~/Desktop/scripts/week7

Ln 36, Col 2



main.c

segment.c

example.c

example2.c



example3.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define SIZE 10
5
6 int main() {
7     int *array = NULL;
8
9     // Allocate memory for an array of integers
10    array = (int *)malloc(SIZE * sizeof(int));
11    if (array == NULL) {
12        printf("Memory allocation failed. Exiting...\n");
13        return 1;
14    }
15
16    // Initialize the array with some values
17    for (int i = 0; i < SIZE; i++) {
18        array[i] = i * 2;
19    }
20
21    // Access elements of the array
22    for (int i = 0; i < SIZE; i++) {
23        printf("array[%d] = %d\n", i, array[i]);
24    }
25
26    // Free the allocated memory
27    free(array);
28
29    // Attempt to access the array after freeing it (causing segmentation fault)
```

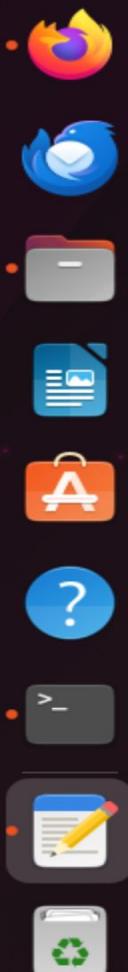
ENG  
IN15:36  
13-03-202415:36  
13-03-2024

PRE

example2.c  
~/Desktop/scripts/week1

Ln 36, Col 2

example3.c





sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7



```
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gcc -g example2.c
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gdb ./a.out
GNU gdb (Ubuntu 13.1-2ubuntu2.1) 13.1
Copyright (C) 2023 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:
<https://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/sridhar/Desktop/scripts/week7/a.out
```

```
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
```



Search

ENG  
IN15:01  
13-03-2024



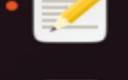
sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

```
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
array[7] = 14
array[8] = 16
array[9] = 18
Trying to access the array after freeing it:
array[0] = 1431655769
array[1] = 5
array[2] = -1462845441
array[3] = -419435296
array[4] = 8
array[5] = 10
array[6] = 12
array[7] = 14
array[8] = 16
array[9] = 18
[Inferior 1 (process 4648) exited normally]
(gdb) break 26
Breakpoint 1 at 0x55555555525b: file example2.c, line 27.
(gdb) run
Starting program: /home/sridhar/Desktop/scripts/week7/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
```



Search

ENG  
IN15:02  
13-03-2024



sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

```
(gdb) run
Starting program: /home/sridhar/Desktop/scripts/week7/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
array[0] = 0
array[1] = 2
array[2] = 4
array[3] = 6
array[4] = 8
array[5] = 10
array[6] = 12
array[7] = 14
array[8] = 16
array[9] = 18

Breakpoint 1, main () at example2.c:27
27          free(array);
(gdb) next
30          printf("Trying to access the array after freeing it:\n");
(gdb) next
Trying to access the array after freeing it:
31          for (int i = 0; i < SIZE; i++) {
(gdb) print[i]
A syntax error in expression, near `'[i]''.
(gdb) print(array[i])
$1 = 1431655769
(gdb) continue
Continuing.
array[0] = 1431655769
array[1] = 5
array[2] = -1012062357
array[3] = -1560134378
array[4] = 8
array[5] = 10
array[6] = 12
```

ENG  
INWi-Fi  
Speaker  
Battery15:02  
13-03-2024Bell  
PRE



```
Trying to access the array after freeing it:  
31      for (int i = 0; i < SIZE; i++) {  
(gdb) print[i]  
A syntax error in expression, near `'[i]'`.  
(gdb) print(array[i])  
$1 = 1431655769  
(gdb) continue  
Continuing.  
array[0] = 1431655769  
array[1] = 5  
array[2] = -1012062357  
array[3] = -1560134378  
array[4] = 8  
array[5] = 10  
array[6] = 12  
array[7] = 14  
array[8] = 16  
array[9] = 18  
[Inferior 1 (process 4656) exited normally]  
(gdb) disassemble main  
Dump of assembler code for function main:  
0x00005555555551a9 <+0>:    endbr64  
0x00005555555551ad <+4>:    push    %rbp  
0x00005555555551ae <+5>:    mov     %rsp,%rbp  
0x00005555555551b1 <+8>:    sub    $0x20,%rsp  
0x00005555555551b5 <+12>:   movq   $0x0,-0x8(%rbp)  
0x00005555555551bd <+20>:   mov    $0x28,%edi  
0x00005555555551c2 <+25>:   call   0x5555555550b0 <malloc@plt>  
0x00005555555551c7 <+30>:   mov    %rax,-0x8(%rbp)  
0x00005555555551cb <+34>:   cmpq   $0x0,-0x8(%rbp)  
0x00005555555551d0 <+39>:   jne    0x5555555551eb <main+66>  
0x00005555555551d2 <+41>:   lea    0xe2f(%rip),%rax        # 0x555555556008  
0x00005555555551d9 <+48>:   mov    %rax,%rdi  
0x00005555555551dc <+51>:   call   0x555555555090 <puts@plt>  
0x00005555555551e1 <+56>:   mov    %rax
```





```
Dump of assembler code for function main:  
0x0000555555551a9 <+0>:    endbr64  
0x0000555555551ad <+4>:    push   %rbp  
0x0000555555551ae <+5>:    mov    %rsp,%rbp  
0x0000555555551b1 <+8>:    sub    $0x20,%rsp  
0x0000555555551b5 <+12>:   movq   $0x0,-0x8(%rbp)  
0x0000555555551bd <+20>:   mov    $0x28,%edi  
0x0000555555551c2 <+25>:   call   0x555555550b0 <malloc@plt>  
0x0000555555551c7 <+30>:   mov    %rax,-0x8(%rbp)  
0x0000555555551cb <+34>:   cmpq   $0x0,-0x8(%rbp)  
0x0000555555551d0 <+39>:   jne    0x555555551eb <main+66>  
0x0000555555551d2 <+41>:   lea    0xe2f(%rip),%rax      # 0x555555556008  
0x0000555555551d9 <+48>:   mov    %rax,%rdi  
0x0000555555551dc <+51>:   call   0x55555555090 <puts@plt>  
0x0000555555551e1 <+56>:   mov    $0x1,%eax  
0x0000555555551e6 <+61>:   jmp    0x555555552bd <main+276>  
0x0000555555551eb <+66>:   movl   $0x0,-0x14(%rbp)  
0x0000555555551f2 <+73>:   jmp    0x55555555213 <main+106>  
0x0000555555551f4 <+75>:   mov    -0x14(%rbp),%eax  
0x0000555555551f7 <+78>:   cltq  
0x0000555555551f9 <+80>:   lea    0x0(%rax,4),%rdx  
0x000055555555201 <+88>:   mov    -0x8(%rbp),%rax  
0x000055555555205 <+92>:   add    %rdx,%rax  
0x000055555555208 <+95>:   mov    -0x14(%rbp),%edx  
0x00005555555520b <+98>:   add    %edx,%edx  
0x00005555555520d <+100>:  mov    %edx,(%rax)  
0x00005555555520f <+102>:  addl   $0x1,-0x14(%rbp)  
0x000055555555213 <+106>:  cmpl   $0x9,-0x14(%rbp)  
0x000055555555217 <+110>:  jle    0x555555551f4 <main+75>  
0x000055555555219 <+112>:  movl   $0x0,-0x10(%rbp)  
0x000055555555220 <+119>:  jmp    0x55555555255 <main+172>  
0x000055555555222 <+121>:  mov    -0x10(%rbp),%eax  
0x000055555555225 <+124>:  cltq
```

-- Type <RET> for more, q to quit, c to continue without paging--





sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7



```
0x000055555555247 <+158>:    mov    $0x0,%eax
0x00005555555524c <+163>:    call   0x555555550a0 <printf@plt>
0x000055555555251 <+168>:    addl   $0x1,-0x10(%rbp)
0x000055555555255 <+172>:    cmpl   $0x9,-0x10(%rbp)
0x000055555555259 <+176>:    jle    0x55555555222 <main+121>
0x00005555555525b <+178>:    mov    -0x8(%rbp),%rax
0x00005555555525f <+182>:    mov    %rax,%rdi
0x000055555555262 <+185>:    call   0x55555555080 <free@plt>
0x000055555555267 <+190>:    lea    0xdd2(%rip),%rax      # 0x555555556040
0x00005555555526e <+197>:    mov    %rax,%rdi
0x000055555555271 <+200>:    call   0x55555555090 <puts@plt>
0x000055555555276 <+205>:    movl   $0x0,-0xc(%rbp)
0x00005555555527d <+212>:    jmp    0x555555552b2 <main+265>
0x00005555555527f <+214>:    mov    -0xc(%rbp),%eax
0x000055555555282 <+217>:    cltq
0x000055555555284 <+219>:    lea    0x0(%rax,4),%rdx
0x00005555555528c <+227>:    mov    -0x8(%rbp),%rax
0x000055555555290 <+231>:    add    %rdx,%rax
0x000055555555293 <+234>:    mov    (%rax),%edx
0x000055555555295 <+236>:    mov    -0xc(%rbp),%eax
0x000055555555298 <+239>:    mov    %eax,%esi
0x00005555555529a <+241>:    lea    0xd8c(%rip),%rax      # 0x55555555602d
0x0000555555552a1 <+248>:    mov    %rax,%rdi
0x0000555555552a4 <+251>:    mov    $0x0,%eax
0x0000555555552a9 <+256>:    call   0x555555550a0 <printf@plt>
--Type <RET> for more, q to quit, c to continue without paging--
0x0000555555552ae <+261>:    addl   $0x1,-0xc(%rbp)
0x0000555555552b2 <+265>:    cmpl   $0x9,-0xc(%rbp)
0x0000555555552b6 <+269>:    jle    0x5555555527f <main+214>
0x0000555555552b8 <+271>:    mov    $0x0,%eax
0x0000555555552bd <+276>:    leave
0x0000555555552be <+277>:    ret
```

End of assembler dump.  
(gdb)



Search

ENG  
IN15:02  
13-03-2024

Single Segmentation fault via Unassigned Pointer with  
multiple breakpoints

ubuntu [Running] - Oracle VM VirtualBox

Activities Text Editor Mar 13 15:36

Open main.c segment.c example.c example2.c example3.c

example3.c  
~/Desktop/scripts/week7 Ln 27, Col 5

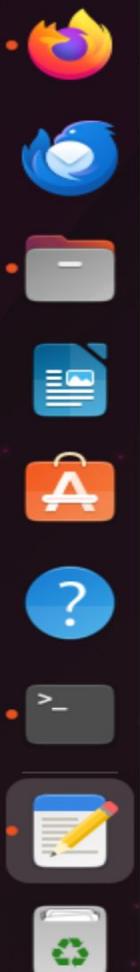
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void access_unallocated_memory() {
5     int *ptr;
6     *ptr = 10; // Attempting to access unallocated memory
7 }
8
9 void do_some_work(int n) {
10    // Simulating some work with a loop
11    int sum = 0;
12    for (int i = 0; i < n; i++) {
13        sum += i;
14    }
15    printf("Sum: %d\n", sum);
16 }
17
18 int main() {
19     printf("Starting program...\n");
20
21     // Simulating some operations
22     for (int i = 0; i < 3; i++) {
23         printf("Iteration %d\n", i);
24         do_some_work(i * 1000);
25     }
26
27     // Attempt: Accessing unallocated memory
28     printf("Attempting to access unallocated memory...\n");
29     access_unallocated_memory(); // This will cause a segmentation fault
```

1 ENG IN 15:36 13-03-2024 PRE

example3.c  
~/Desktop/scripts/week1

Ln 27, Col 5

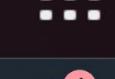
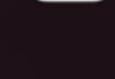
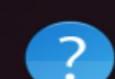
example3.c



Activities Terminal

Mar 13 15:31

Speaker Battery



sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7

Search Minimize Maximize Close

```
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gcc -g example3.c
sridhar@sridhar-VirtualBox:~/Desktop/scripts/week7$ gdb ./a.out
GNU gdb (Ubuntu 13.1-2ubuntu2.1) 13.1
Copyright (C) 2023 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:
<https://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/sridhar/Desktop/scripts/week7/a.out
```

```
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y\
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Starting program...
```

```
Iteration 0
Sum: 0
Iteration 1
Sum: 499500
Iteration 2
Sum: 1000000
```



Search



ENG IN

15:31 13-03-2024

15:31 13-03-2024

PRE

```
Sum: 499500
Iteration 2
Sum: 1999000
Attempting to access unallocated memory...
Program continues after the segmentation fault.
[Inferior 1 (process 5025) exited normally]
(gdb) list
5         int *ptr;
6             *ptr = 10; // Attempting to access unallocated memory
7
8
9     void do_some_work(int n) {
10        // Simulating some work with a loop
11        int sum = 0;
12        for (int i = 0; i < n; i++) {
13            sum += i;
14        }
(gdb) break 4
Breakpoint 1 at 0x555555555171: file example3.c, line 6.
(gdb) break 15
Breakpoint 2 at 0x5555555551af: file example3.c, line 15.
(gdb) run
Starting program: /home/sridhar/Desktop/scripts/week7/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Starting program...
Iteration 0

Breakpoint 2, do_some_work (n=0) at example3.c:15
15         printf("Sum: %d\n", sum);
(gdb) next
Sum: 0
16
(gdb) next
main () at example3.c:22
```





```
16      }
(gdb) next
main () at example3.c:22
22      for (int i = 0; i < 3; i++) {
(gdb) continue
Continuing.
Iteration 1
```



```
Breakpoint 2, do_some_work (n=1000) at example3.c:15
15      printf("Sum: %d\n", sum);
```



```
(gdb) continue
Continuing.
Sum: 499500
Iteration 2
```



```
Breakpoint 2, do_some_work (n=2000) at example3.c:15
15      printf("Sum: %d\n", sum);
```



```
(gdb) continue
Continuing.
Sum: 1999000
Attempting to access unallocated memory...
```



```
Breakpoint 1, access_unallocated_memory () at example3.c:6
6      *ptr = 10; // Attempting to access unallocated memory
```



```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sridhar/Desktop/scripts/week7/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Starting program...
Iteration 0
```



```
Breakpoint 2, do_some_work (n=0) at example3.c:15
15      printf("Sum: %d\n", sum);
```





J+



```
Breakpoint 2, do_some_work (n=0) at example3.c:15
15      printf("Sum: %d\n", sum);
```



```
(gdb) continue
Continuing.
Sum: 0
Iteration 1
```



```
Breakpoint 2, do_some_work (n=1000) at example3.c:15
15      printf("Sum: %d\n", sum);
```



```
(gdb) break 26
Breakpoint 3 at 0x555555555222: file example3.c, line 28.
(gdb) continue
Continuing.
```



```
Sum: 499500
Iteration 2
```



```
Breakpoint 2, do_some_work (n=2000) at example3.c:15
15      printf("Sum: %d\n", sum);
```



```
(gdb) next
Sum: 1999000
16 }
```



```
(gdb) next
main () at example3.c:22
22      for (int i = 0; i < 3; i++) {
(gdb) continue
Continuing.
```



```
Breakpoint 3, main () at example3.c:28
28      printf("Attempting to access unallocated memory...\n");
```



Search



^



ENG

IN

15:31  
13-03-2024

PRE



```
Breakpoint 3, main () at example3.c:28
28         printf("Attempting to access unallocated memory...\n");
(gdb) continue
Continuing.
Attempting to access unallocated memory...

Breakpoint 1, access_unallocated_memory () at example3.c:6
6         *ptr = 10; // Attempting to access unallocated memory
(gdb) disassemble main
Dump of assembler code for function main:
0x0000555555551cb <+0>:    endbr64
0x0000555555551cf <+4>:    push   %rbp
0x0000555555551d0 <+5>:    mov    %rsp,%rbp
0x0000555555551d3 <+8>:    sub    $0x10,%rsp
0x0000555555551d7 <+12>:   lea    0xe33(%rip),%rax      # 0x555555556011
0x0000555555551de <+19>:   mov    %rax,%rdi
0x0000555555551e1 <+22>:   call   0x55555555060 <puts@plt>
0x0000555555551e6 <+27>:   movl   $0x0,-0x4(%rbp)
0x0000555555551ed <+34>:   jmp   0x5555555521c <main+81>
0x0000555555551ef <+36>:   mov    -0x4(%rbp),%eax
0x0000555555551f2 <+39>:   mov    %eax,%esi
0x0000555555551f4 <+41>:   lea    0xe2a(%rip),%rax      # 0x555555556025
0x0000555555551fb <+48>:   mov    %rax,%rdi
0x0000555555551fe <+51>:   mov    $0x0,%eax
0x000055555555203 <+56>:   call   0x55555555070 <printf@plt>
0x000055555555208 <+61>:   mov    -0x4(%rbp),%eax
0x00005555555520b <+64>:   imul   $0x3e8,%eax,%eax
0x000055555555211 <+70>:   mov    %eax,%edi
0x000055555555213 <+72>:   call   0x5555555517e <do_some_work>
0x000055555555218 <+77>:   addl   $0x1,-0x4(%rbp)
0x00005555555521c <+81>:   cmpl   $0x2,-0x4(%rbp)
0x000055555555220 <+85>:   jle    0x555555551ef <main+36>
0x000055555555222 <+87>:   lea    0xe0f(%rip),%rax      # 0x555555556038
0x000055555555229 <+94>:   mov    %rax,%rdi
0x00005555555522d <+97>:   callq  0x55555555060 <puts@plt>
```



Search

ENG  
IN15:31  
13-03-2024



sridhar@sridhar-VirtualBox: ~/Desktop/scripts/week7



```
0x0000555555551d0 <+5>:    mov    %rsp,%rbp
0x0000555555551d3 <+8>:    sub    $0x10,%rsp
0x0000555555551d7 <+12>:   lea    0xe33(%rip),%rax      # 0x555555556011
0x0000555555551de <+19>:   mov    %rax,%rdi
0x0000555555551e1 <+22>:   call   0x55555555060 <puts@plt>
0x0000555555551e6 <+27>:   movl   $0x0,-0x4(%rbp)
0x0000555555551ed <+34>:   jmp    0x5555555521c <main+81>
0x0000555555551ef <+36>:   mov    -0x4(%rbp),%eax
0x0000555555551f2 <+39>:   mov    %eax,%esi
0x0000555555551f4 <+41>:   lea    0xe2a(%rip),%rax      # 0x555555556025
0x0000555555551fb <+48>:   mov    %rax,%rdi
0x0000555555551fe <+51>:   mov    $0x0,%eax
0x000055555555203 <+56>:   call   0x55555555070 <printf@plt>
0x000055555555208 <+61>:   mov    -0x4(%rbp),%eax
0x00005555555520b <+64>:   imul   $0x3e8,%eax,%eax
0x000055555555211 <+70>:   mov    %eax,%edi
0x000055555555213 <+72>:   call   0x5555555517e <do_some_work>
0x000055555555218 <+77>:   addl   $0x1,-0x4(%rbp)
0x00005555555521c <+81>:   cmpl   $0x2,-0x4(%rbp)
0x000055555555220 <+85>:   jle    0x555555551ef <main+36>
0x000055555555222 <+87>:   lea    0xe0f(%rip),%rax      # 0x555555556038
0x000055555555229 <+94>:   mov    %rax,%rdi
0x00005555555522c <+97>:   call   0x55555555060 <puts@plt>
0x000055555555231 <+102>:  mov    $0x0,%eax
0x000055555555236 <+107>:  call   0x55555555169 <access_unallocated_memory>
0x00005555555523b <+112>:  lea    0xe26(%rip),%rax      # 0x555555556068
0x000055555555242 <+119>:  mov    %rax,%rdi
0x000055555555245 <+122>:  call   0x55555555060 <puts@plt>
0x00005555555524a <+127>:  mov    $0x0,%eax
0x00005555555524f <+132>:  leave 
--Type <RET> for more, q to quit, c to continue without paging--
0x000055555555250 <+133>:  ret

```

End of assembler dump.  
(gdb) █



Search

ENG  
IN15:31  
13-03-2024

Work by:  
K.Sridhar Varma(422165)