Unix week-9
GNU Debugger

Program showing NULL pointer access.

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

#define SIZE 10

int main() {
    int *array = NULL;

    // Allocate memory for an array of integers
    array = (int *)malloc(SIZE * sizeof(int));
    if (array == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }

    // Initialize the array with some values
    for (int i = 0; i < SIZE; i++) {
        array[i] = i * 2;
    }

    // Access elements of the array
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    // Free the allocated memory
    array = NULL;

    // Attempt to access the array after freeing it (causing segmentation fault)
```

```c
    // Access elements of the array
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    // Free the allocated memory
    array = NULL;

    // Attempt to access the array after freeing it (causing segmentation fault)
    printf("Trying to access the array after freeing it:\n");
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    return 0;
}
```

```
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
```

Show Apps

```
array[8] = 16
array[9] = 18
Trying to access the array after freeing it:

Program received signal SIGSEGV, Segmentation fault.
0x000055555555526f 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 0x55555555523b: 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[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.
0x000055555555526f 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:
   0x0000555555555189 <+0>:     endbr64
   0x000055555555518d <+4>:     push   %rbp
   0x000055555555518e <+5>:     mov    %rsp,%rbp
   0x0000555555555191 <+8>:     sub    $0x20,%rsp
   0x0000555555555195 <+12>:    movq   $0x0,-0x8(%rbp)
   0x000055555555519d <+20>:    mov    $0x28,%edi
   0x00005555555551a2 <+25>:    call   0x555555555090 <malloc@plt>
   0x00005555555551a7 <+30>:    mov    %rax,-0x8(%rbp)
   0x00005555555551ab <+34>:    cmpq   $0x0,-0x8(%rbp)
   0x00005555555551b0 <+39>:    jne    0x5555555551cb <main+66>
   0x00005555555551b2 <+41>:    lea    0xe4f(%rip),%rax        # 0x555555556008
   0x00005555555551b9 <+48>:    mov    %rax,%rdi
   0x00005555555551bc <+51>:    call   0x555555555070 <puts@plt>
   0x00005555555551c1 <+56>:    mov    $0x1,%eax
```

```
   0x0000555555555189 <+0>:     endbr64
   0x000055555555518d <+4>:     push   %rbp
   0x000055555555518e <+5>:     mov    %rsp,%rbp
   0x0000555555555191 <+8>:     sub    $0x20,%rsp
   0x0000555555555195 <+12>:    movq   $0x0,-0x8(%rbp)
   0x000055555555519d <+20>:    mov    $0x28,%edi
   0x00005555555551a2 <+25>:    call   0x555555555090 <malloc@plt>
   0x00005555555551a7 <+30>:    mov    %rax,-0x8(%rbp)
   0x00005555555551ab <+34>:    cmpq   $0x0,-0x8(%rbp)
   0x00005555555551b0 <+39>:    jne    0x5555555551cb <main+66>
   0x00005555555551b2 <+41>:    lea    0xe4f(%rip),%rax        # 0x555555556008
   0x00005555555551b9 <+48>:    mov    %rax,%rdi
   0x00005555555551bc <+51>:    call   0x555555555070 <puts@plt>
   0x00005555555551c1 <+56>:    mov    $0x1,%eax
   0x00005555555551c6 <+61>:    jmp    0x555555555299 <main+272>
   0x00005555555551cb <+66>:    movl   $0x0,-0x14(%rbp)
   0x00005555555551d2 <+73>:    jmp    0x5555555551f3 <main+106>
   0x00005555555551d4 <+75>:    mov    -0x14(%rbp),%eax
   0x00005555555551d7 <+78>:    cltq
   0x00005555555551d9 <+80>:    lea    0x0(,%rax,4),%rdx
   0x00005555555551e1 <+88>:    mov    -0x8(%rbp),%rax
   0x00005555555551e5 <+92>:    add    %rdx,%rax
   0x00005555555551e8 <+95>:    mov    -0x14(%rbp),%edx
   0x00005555555551eb <+98>:    add    %edx,%edx
   0x00005555555551ed <+100>:   mov    %edx,(%rax)
   0x00005555555551ef <+102>:   addl   $0x1,-0x14(%rbp)
   0x00005555555551f3 <+106>:   cmpl   $0x9,-0x14(%rbp)
   0x00005555555551f7 <+110>:   jle    0x5555555551d4 <main+75>
   0x00005555555551f9 <+112>:   movl   $0x0,-0x10(%rbp)
   0x0000555555555200 <+119>:   jmp    0x555555555235 <main+172>
   0x0000555555555202 <+121>:   mov    -0x10(%rbp),%eax
   0x0000555555555205 <+124>:   cltq
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000555555555207 <+126>:   lea    0x0(,%rax,4),%rdx
```

```
   0x0000555555555202 <+121>:   mov    -0x10(%rbp),%eax
   0x0000555555555205 <+124>:   cltq
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000555555555207 <+126>:   lea    0x0(,%rax,4),%rdx
   0x000055555555520f <+134>:   mov    -0x8(%rbp),%rax
   0x0000555555555213 <+138>:   add    %rdx,%rax
   0x0000555555555216 <+141>:   mov    (%rax),%edx
   0x0000555555555218 <+143>:   mov    -0x10(%rbp),%eax
   0x000055555555521b <+146>:   mov    %eax,%esi
   0x000055555555521d <+148>:   lea    0xe09(%rip),%rax        # 0x55555555602d
   0x0000555555555224 <+155>:   mov    %rax,%rdi
   0x0000555555555227 <+158>:   mov    $0x0,%eax
   0x000055555555522c <+163>:   call   0x555555555080 <printf@plt>
   0x0000555555555231 <+168>:   addl   $0x1,-0x10(%rbp)
   0x0000555555555235 <+172>:   cmpl   $0x9,-0x10(%rbp)
   0x0000555555555239 <+176>:   jle    0x555555555202 <main+121>
   0x000055555555523b <+178>:   movq   $0x0,-0x8(%rbp)
   0x0000555555555243 <+186>:   lea    0xdf6(%rip),%rax        # 0x555555556040
   0x000055555555524a <+193>:   mov    %rax,%rdi
   0x000055555555524d <+196>:   call   0x555555555070 <puts@plt>
   0x0000555555555252 <+201>:   movl   $0x0,-0xc(%rbp)
   0x0000555555555259 <+208>:   jmp    0x55555555528e <main+261>
   0x000055555555525b <+210>:   mov    -0xc(%rbp),%eax
   0x000055555555525e <+213>:   cltq
   0x0000555555555260 <+215>:   lea    0x0(,%rax,4),%rdx
   0x0000555555555268 <+223>:   mov    -0x8(%rbp),%rax
   0x000055555555526c <+227>:   add    %rdx,%rax
   0x000055555555526f <+230>:   mov    (%rax),%edx
   0x0000555555555271 <+232>:   mov    -0xc(%rbp),%eax
   0x0000555555555274 <+235>:   mov    %eax,%esi
   0x0000555555555276 <+237>:   lea    0xdb0(%rip),%rax        # 0x55555555602d
   0x000055555555527d <+244>:   mov    %rax,%rdi
   0x0000555555555280 <+247>:   mov    $0x0,%eax
   0x0000555555555285 <+252>:   call   0x555555555080 <printf@plt>
```

```
   0x000055555555521b <+146>:   mov    %eax,%esi
   0x000055555555521d <+148>:   lea    0xe09(%rip),%rax        # 0x55555555602d
   0x0000555555555224 <+155>:   mov    %rax,%rdi
   0x0000555555555227 <+158>:   mov    $0x0,%eax
   0x000055555555522c <+163>:   call   0x555555555080 <printf@plt>
   0x0000555555555231 <+168>:   addl   $0x1,-0x10(%rbp)
   0x0000555555555235 <+172>:   cmpl   $0x9,-0x10(%rbp)
   0x0000555555555239 <+176>:   jle    0x555555555202 <main+121>
   0x000055555555523b <+178>:   movq   $0x0,-0x8(%rbp)
   0x0000555555555243 <+186>:   lea    0xdf6(%rip),%rax        # 0x555555556040
   0x000055555555524a <+193>:   mov    %rax,%rdi
   0x000055555555524d <+196>:   call   0x555555555070 <puts@plt>
   0x0000555555555252 <+201>:   movl   $0x0,-0xc(%rbp)
   0x0000555555555259 <+208>:   jmp    0x55555555528e <main+261>
   0x000055555555525b <+210>:   mov    -0xc(%rbp),%eax
   0x000055555555525e <+213>:   cltq
   0x0000555555555260 <+215>:   lea    0x0(,%rax,4),%rdx
   0x0000555555555268 <+223>:   mov    -0x8(%rbp),%rax
   0x000055555555526c <+227>:   add    %rdx,%rax
   0x000055555555526f <+230>:   mov    (%rax),%edx
   0x0000555555555271 <+232>:   mov    -0xc(%rbp),%eax
   0x0000555555555274 <+235>:   mov    %eax,%esi
   0x0000555555555276 <+237>:   lea    0xdb0(%rip),%rax        # 0x55555555602d
   0x000055555555527d <+244>:   mov    %rax,%rdi
   0x0000555555555280 <+247>:   mov    $0x0,%eax
   0x0000555555555285 <+252>:   call   0x555555555080 <printf@plt>
   0x000055555555528a <+257>:   addl   $0x1,-0xc(%rbp)
   0x000055555555528e <+261>:   cmpl   $0x9,-0xc(%rbp)
   0x0000555555555292 <+265>:   jle    0x55555555525b <main+210>
   0x0000555555555294 <+267>:   mov    $0x0,%eax
   0x0000555555555299 <+272>:   leave
   0x000055555555529a <+273>:   ret
End of assembler dump.
(gdb)
```

Illegal Memory Access

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

#define SIZE 10

int main() {
    int *array = NULL;

    // Allocate memory for an array of integers
    array = (int *)malloc(SIZE * sizeof(int));
    if (array == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }

    // Initialize the array with some values
    for (int i = 0; i < SIZE; i++) {
        array[i] = i * 2;
    }

    // Access elements of the array
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    // Free the allocated memory
    free(array);

    // Attempt to access the array after freeing it (causing segmentation fault)
```

```c
    for (int i = 0; i < SIZE; i++) {
        array[i] = i * 2;
    }

    // Access elements of the array
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    // Free the allocated memory
    free(array);

    // Attempt to access the array after freeing it (causing segmentation fault)
    printf("Trying to access the array after freeing it:\n");
    for (int i = 0; i < SIZE; i++) {
        printf("array[%d] = %d\n", i, array[i]);
    }

    return 0;
}
```

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
```

Show Apps

```
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
```

```
(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
```

```
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>
```

```
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    $0x1,%eax
   0x00005555555551e6 <+61>:    jmp    0x5555555552bd <main+276>
   0x00005555555551eb <+66>:    movl   $0x0,-0x14(%rbp)
   0x00005555555551f2 <+73>:    jmp    0x555555555213 <main+106>
   0x00005555555551f4 <+75>:    mov    -0x14(%rbp),%eax
   0x00005555555551f7 <+78>:    cltq
   0x00005555555551f9 <+80>:    lea    0x0(,%rax,4),%rdx
   0x0000555555555201 <+88>:    mov    -0x8(%rbp),%rax
   0x0000555555555205 <+92>:    add    %rdx,%rax
   0x0000555555555208 <+95>:    mov    -0x14(%rbp),%edx
   0x000055555555520b <+98>:    add    %edx,%edx
   0x000055555555520d <+100>:   mov    %edx,(%rax)
   0x000055555555520f <+102>:   addl   $0x1,-0x14(%rbp)
   0x0000555555555213 <+106>:   cmpl   $0x9,-0x14(%rbp)
   0x0000555555555217 <+110>:   jle    0x5555555551f4 <main+75>
   0x0000555555555219 <+112>:   movl   $0x0,-0x10(%rbp)
   0x0000555555555220 <+119>:   jmp    0x555555555255 <main+172>
   0x0000555555555222 <+121>:   mov    -0x10(%rbp),%eax
   0x0000555555555225 <+124>:   cltq
--Type <RET> for more, q to quit, c to continue without paging--
```

```
   0x0000555555555247 <+158>:   mov    $0x0,%eax
   0x000055555555524c <+163>:   call   0x5555555550a0 <printf@plt>
   0x0000555555555251 <+168>:   addl   $0x1,-0x10(%rbp)
   0x0000555555555255 <+172>:   cmpl   $0x9,-0x10(%rbp)
   0x0000555555555259 <+176>:   jle    0x555555555222 <main+121>
   0x000055555555525b <+178>:   mov    -0x8(%rbp),%rax
   0x000055555555525f <+182>:   mov    %rax,%rdi
   0x0000555555555262 <+185>:   call   0x555555555080 <free@plt>
   0x0000555555555267 <+190>:   lea    0xdd2(%rip),%rax        # 0x555555556040
   0x000055555555526e <+197>:   mov    %rax,%rdi
   0x0000555555555271 <+200>:   call   0x555555555090 <puts@plt>
   0x0000555555555276 <+205>:   movl   $0x0,-0xc(%rbp)
   0x000055555555527d <+212>:   jmp    0x5555555552b2 <main+265>
   0x000055555555527f <+214>:   mov    -0xc(%rbp),%eax
   0x0000555555555282 <+217>:   cltq
   0x0000555555555284 <+219>:   lea    0x0(,%rax,4),%rdx
   0x000055555555528c <+227>:   mov    -0x8(%rbp),%rax
   0x0000555555555290 <+231>:   add    %rdx,%rax
   0x0000555555555293 <+234>:   mov    (%rax),%edx
   0x0000555555555295 <+236>:   mov    -0xc(%rbp),%eax
   0x0000555555555298 <+239>:   mov    %eax,%esi
   0x000055555555529a <+241>:   lea    0xd8c(%rip),%rax        # 0x55555555602d
   0x00005555555552a1 <+248>:   mov    %rax,%rdi
   0x00005555555552a4 <+251>:   mov    $0x0,%eax
   0x00005555555552a9 <+256>:   call   0x5555555550a0 <printf@plt>
--Type <RET> for more, q to quit, c to continue without paging--
   0x00005555555552ae <+261>:   addl   $0x1,-0xc(%rbp)
   0x00005555555552b2 <+265>:   cmpl   $0x9,-0xc(%rbp)
   0x00005555555552b6 <+269>:   jle    0x55555555527f <main+214>
   0x00005555555552b8 <+271>:   mov    $0x0,%eax
   0x00005555555552bd <+276>:   leave
   0x00005555555552be <+277>:   ret
End of assembler dump.
(gdb)
```

Single Segmentation fault via Unassigned Pointer with multiple breakpoints

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

main.c | segment.c | example.c | example2.c | example3.c

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

void access_unallocated_memory() {
    int *ptr;
    *ptr = 10; // Attempting to access unallocated memory
}

void do_some_work(int n) {
    // Simulating some work with a loop
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += i;
    }
    printf("Sum: %d\n", sum);
}

int main() {
    printf("Starting program...\n");

    // Simulating some operations
    for (int i = 0; i < 3; i++) {
        printf("Iteration %d\n", i);
        do_some_work(i * 1000);
    }

    // Attempt: Accessing unallocated memory
    printf("Attempting to access unallocated memory...\n");
    access_unallocated_memory(); // This will cause a segmentation fault
```

```c
    }
    printf("Sum: %d\n", sum);
}

int main() {
    printf("Starting program...\n");

    // Simulating some operations
    for (int i = 0; i < 3; i++) {
        printf("Iteration %d\n", i);
        do_some_work(i * 1000);
    }

    // Attempt: Accessing unallocated memory
    printf("Attempting to access unallocated memory...\n");
    access_unallocated_memory(); // This will cause a segmentation fault

    // More operations after the segmentation fault
    printf("Program continues after the segmentation fault.\n");

    return 0;
}
```

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

```
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: 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
```

```
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
```

```
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");
(gdb) continue
Continuing.
Attempting to access unallocated memory...
```

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

```
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:
   0x00005555555551cb <+0>:     endbr64
   0x00005555555551cf <+4>:     push   %rbp
   0x00005555555551d0 <+5>:     mov    %rsp,%rbp
   0x00005555555551d3 <+8>:     sub    $0x10,%rsp
   0x00005555555551d7 <+12>:    lea    0xe33(%rip),%rax        # 0x555555556011
   0x00005555555551de <+19>:    mov    %rax,%rdi
   0x00005555555551e1 <+22>:    call   0x555555555060 <puts@plt>
   0x00005555555551e6 <+27>:    movl   $0x0,-0x4(%rbp)
   0x00005555555551ed <+34>:    jmp    0x55555555521c <main+81>
   0x00005555555551ef <+36>:    mov    -0x4(%rbp),%eax
   0x00005555555551f2 <+39>:    mov    %eax,%esi
   0x00005555555551f4 <+41>:    lea    0xe2a(%rip),%rax        # 0x555555556025
   0x00005555555551fb <+48>:    mov    %rax,%rdi
   0x00005555555551fe <+51>:    mov    $0x0,%eax
   0x0000555555555203 <+56>:    call   0x555555555070 <printf@plt>
   0x0000555555555208 <+61>:    mov    -0x4(%rbp),%eax
   0x000055555555520b <+64>:    imul   $0x3e8,%eax,%eax
   0x0000555555555211 <+70>:    mov    %eax,%edi
   0x0000555555555213 <+72>:    call   0x55555555517e <do_some_work>
   0x0000555555555218 <+77>:    addl   $0x1,-0x4(%rbp)
   0x000055555555521c <+81>:    cmpl   $0x2,-0x4(%rbp)
   0x0000555555555220 <+85>:    jle    0x5555555551ef <main+36>
   0x0000555555555222 <+87>:    lea    0xe0f(%rip),%rax        # 0x555555556038
   0x0000555555555229 <+94>:    mov    %rax,%rdi
```

```
   0x00005555555551d0 <+5>:     mov    %rsp,%rbp
   0x00005555555551d3 <+8>:     sub    $0x10,%rsp
   0x00005555555551d7 <+12>:    lea    0xe33(%rip),%rax        # 0x555555556011
   0x00005555555551de <+19>:    mov    %rax,%rdi
   0x00005555555551e1 <+22>:    call   0x555555555060 <puts@plt>
   0x00005555555551e6 <+27>:    movl   $0x0,-0x4(%rbp)
   0x00005555555551ed <+34>:    jmp    0x55555555521c <main+81>
   0x00005555555551ef <+36>:    mov    -0x4(%rbp),%eax
   0x00005555555551f2 <+39>:    mov    %eax,%esi
   0x00005555555551f4 <+41>:    lea    0xe2a(%rip),%rax        # 0x555555556025
   0x00005555555551fb <+48>:    mov    %rax,%rdi
   0x00005555555551fe <+51>:    mov    $0x0,%eax
   0x0000555555555203 <+56>:    call   0x555555555070 <printf@plt>
   0x0000555555555208 <+61>:    mov    -0x4(%rbp),%eax
   0x000055555555520b <+64>:    imul   $0x3e8,%eax,%eax
   0x0000555555555211 <+70>:    mov    %eax,%edi
   0x0000555555555213 <+72>:    call   0x55555555517e <do_some_work>
   0x0000555555555218 <+77>:    addl   $0x1,-0x4(%rbp)
   0x000055555555521c <+81>:    cmpl   $0x2,-0x4(%rbp)
   0x0000555555555220 <+85>:    jle    0x5555555551ef <main+36>
   0x0000555555555222 <+87>:    lea    0xe0f(%rip),%rax        # 0x555555556038
   0x0000555555555229 <+94>:    mov    %rax,%rdi
   0x000055555555522c <+97>:    call   0x555555555060 <puts@plt>
   0x0000555555555231 <+102>:   mov    $0x0,%eax
   0x0000555555555236 <+107>:   call   0x555555555169 <access_unallocated_memory>
   0x000055555555523b <+112>:   lea    0xe26(%rip),%rax        # 0x555555556068
   0x0000555555555242 <+119>:   mov    %rax,%rdi
   0x0000555555555245 <+122>:   call   0x555555555060 <puts@plt>
   0x000055555555524a <+127>:   mov    $0x0,%eax
   0x000055555555524f <+132>:   leave
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000555555555250 <+133>:   ret
End of assembler dump.
(gdb)
```

Work by:

K.Sridhar Varma(422165)