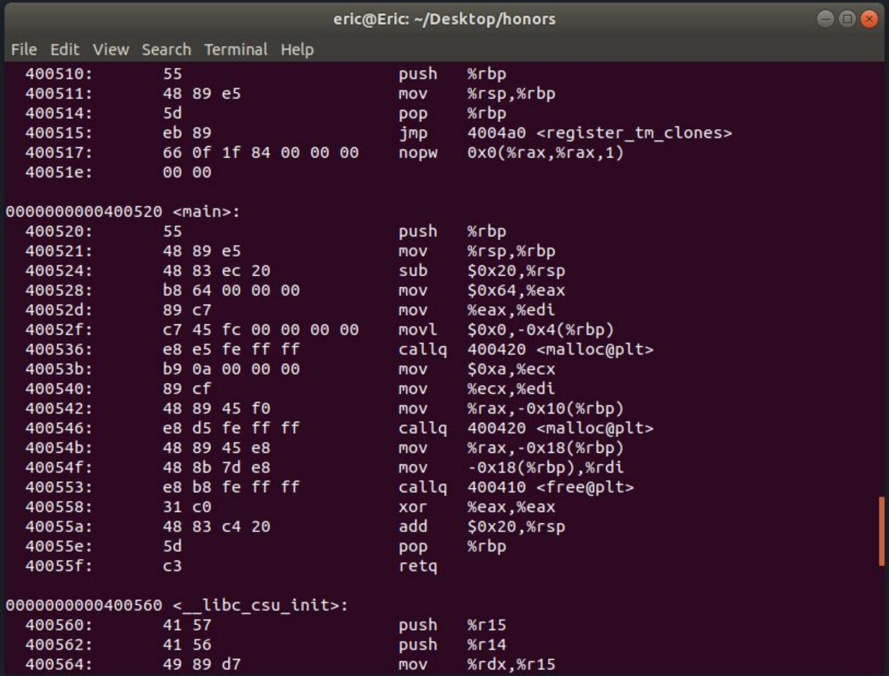
# CS 296 - 41 Honors Final Presentation

Wei-Chen (Eric) Wang

# Topic

- Data visualization of variables declared throughout the program lifespan
- Retrieving variable names for easier debug
- Array library which prevents memory corruption and supports visualization

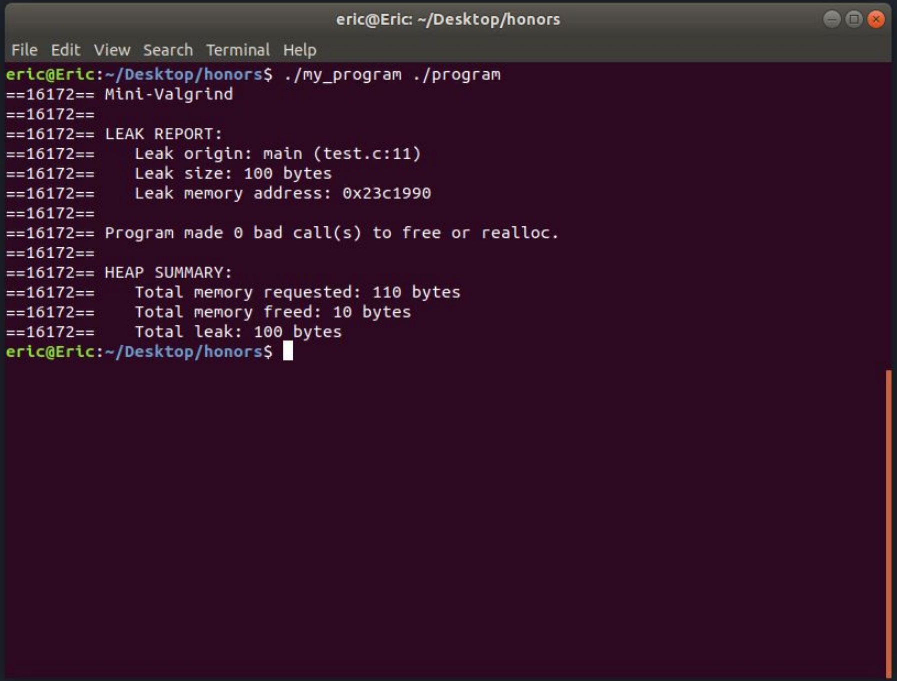
# Research 1: objdump -d



eric@Eric: ~/Desktop/honors
File Edit View Search Terminal Help

```
  400510:       55                      push   %rbp
  400511:       48 89 e5                mov    %rsp,%rbp
  400514:       5d                      pop    %rbp
  400515:       eb 89                   jmp    4004a0 <register_tm_clones>
  400517:       66 0f 1f 84 00 00 00    nopw   0x0(%rax,%rax,1)
  40051e:       00 00

0000000000400520 <main>:
  400520:       55                      push   %rbp
  400521:       48 89 e5                mov    %rsp,%rbp
  400524:       48 83 ec 20             sub    $0x20,%rsp
  400528:       b8 64 00 00 00          mov    $0x64,%eax
  40052d:       89 c7                   mov    %eax,%edi
  40052f:       c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  400536:       e8 e5 fe ff ff          callq  400420 <malloc@plt>
  40053b:       b9 0a 00 00 00          mov    $0xa,%ecx
  400540:       89 cf                   mov    %ecx,%edi
  400542:       48 89 45 f0             mov    %rax,-0x10(%rbp)
  400546:       e8 d5 fe ff ff          callq  400420 <malloc@plt>
  40054b:       48 89 45 e8             mov    %rax,-0x18(%rbp)
  40054f:       48 8b 7d e8             mov    -0x18(%rbp),%rdi
  400553:       e8 b8 fe ff ff          callq  400410 <free@plt>
  400558:       31 c0                   xor    %eax,%eax
  40055a:       48 83 c4 20             add    $0x20,%rsp
  40055e:       5d                      pop    %rbp
  40055f:       c3                      retq

0000000000400560 <__libc_csu_init>:
  400560:       41 57                   push   %r15
  400562:       41 56                   push   %r14
  400564:       49 89 d7                mov    %rdx,%r15
```

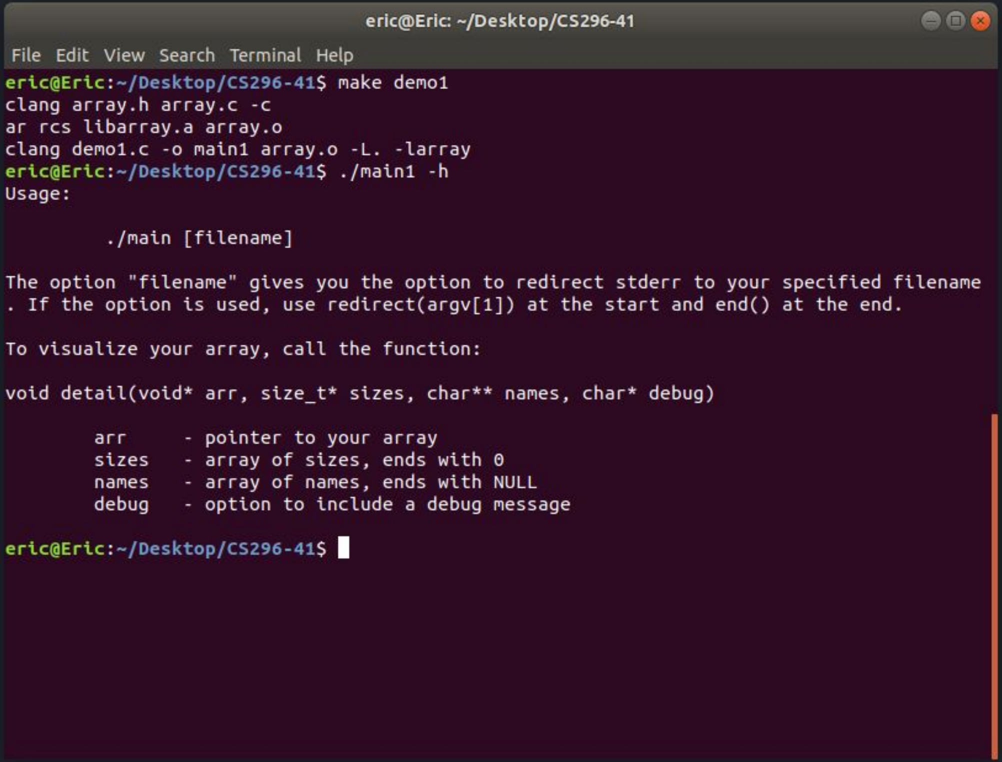
# Research 2: How valgrind (and hooks) works



eric@Eric: ~/Desktop/honors
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/honors$ ./my_program ./program
==16172== Mini-Valgrind
==16172==
==16172== LEAK REPORT:
==16172==    Leak origin: main (test.c:11)
==16172==    Leak size: 100 bytes
==16172==    Leak memory address: 0x23c1990
==16172==
==16172== Program made 0 bad call(s) to free or realloc.
==16172==
==16172== HEAP SUMMARY:
==16172==    Total memory requested: 110 bytes
==16172==    Total memory freed: 10 bytes
==16172==    Total leak: 100 bytes
eric@Eric:~/Desktop/honors$
```

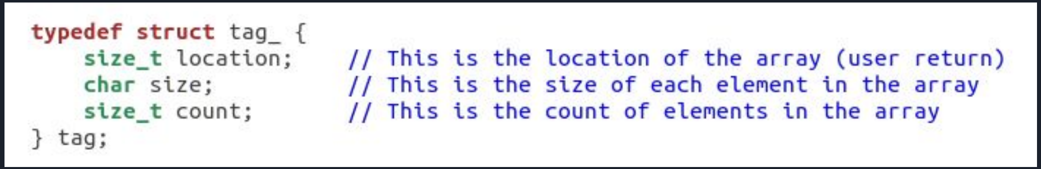
# Array usage



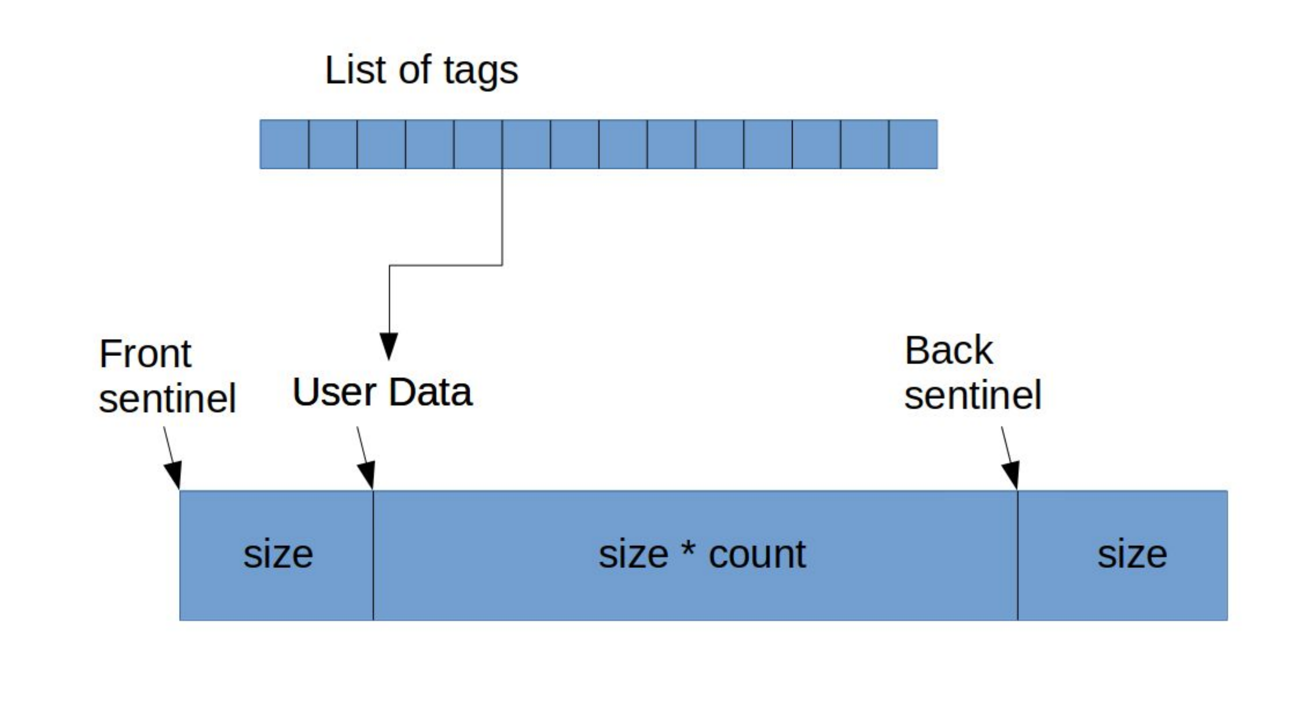
eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/CS296-41$ make demo1
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo1.c -o main1 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main1 -h
Usage:

         ./main [filename]

The option "filename" gives you the option to redirect stderr to your specified filename
. If the option is used, use redirect(argv[1]) at the start and end() at the end.

To visualize your array, call the function:

void detail(void* arr, size_t* sizes, char** names, char* debug)

        arr      - pointer to your array
        sizes    - array of sizes, ends with 0
        names    - array of names, ends with NULL
        debug    - option to include a debug message

eric@Eric:~/Desktop/CS296-41$ 
```

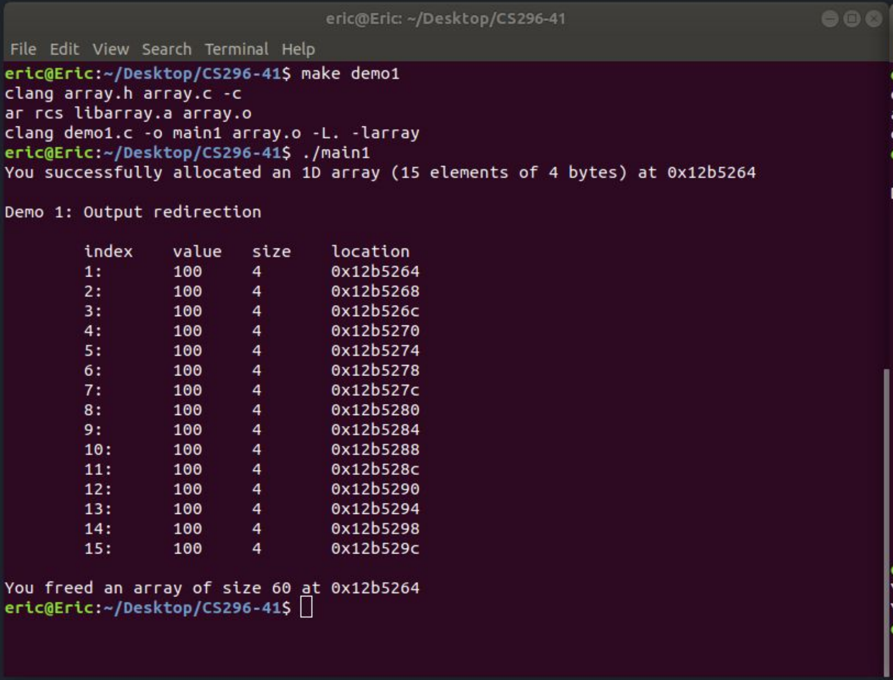
# Array structure



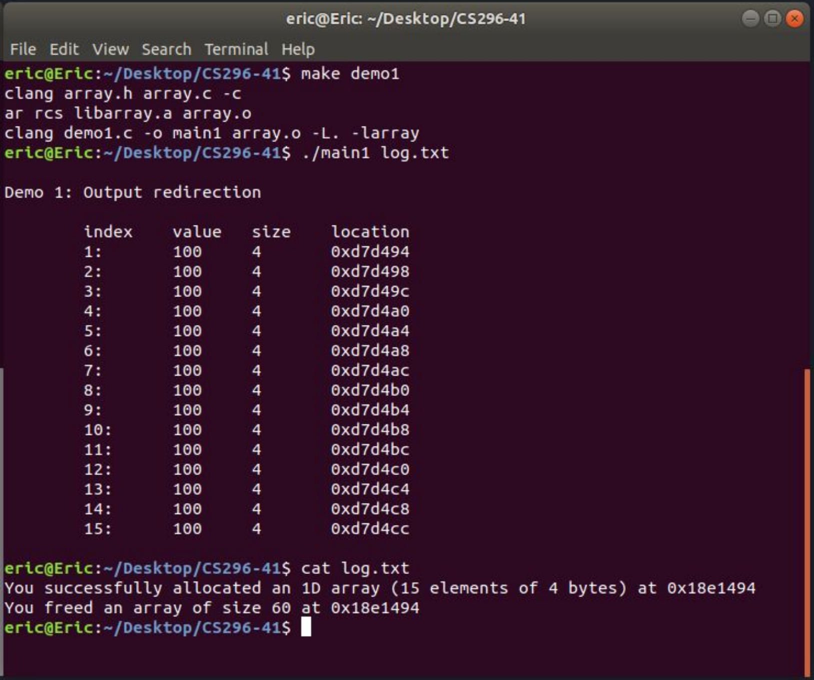
```
typedef struct tag_ {
    size_t location;    // This is the location of the array (user return)
    char size;          // This is the size of each element in the array
    size_t count;       // This is the count of elements in the array
} tag;
```



# Support 1: output redirection


eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

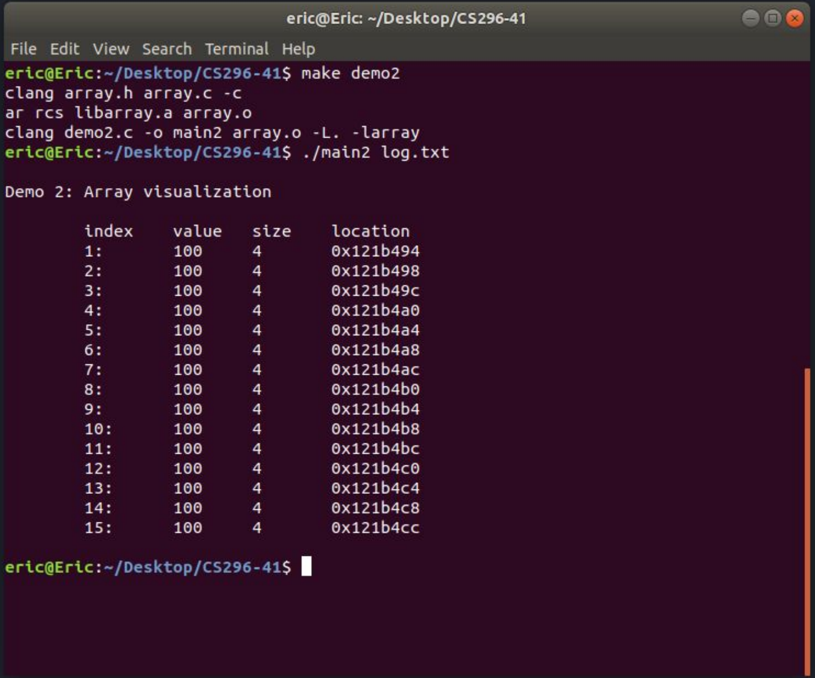
```
eric@Eric:~/Desktop/CS296-41$ make demo1
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo1.c -o main1 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main1
You successfully allocated an 1D array (15 elements of 4 bytes) at 0x12b5264

Demo 1: Output redirection

        index    value   size    location
        1:       100     4       0x12b5264
        2:       100     4       0x12b5268
        3:       100     4       0x12b526c
        4:       100     4       0x12b5270
        5:       100     4       0x12b5274
        6:       100     4       0x12b5278
        7:       100     4       0x12b527c
        8:       100     4       0x12b5280
        9:       100     4       0x12b5284
        10:      100     4       0x12b5288
        11:      100     4       0x12b528c
        12:      100     4       0x12b5290
        13:      100     4       0x12b5294
        14:      100     4       0x12b5298
        15:      100     4       0x12b529c

You freed an array of size 60 at 0x12b5264
eric@Eric:~/Desktop/CS296-41$
```


eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/CS296-41$ make demo1
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo1.c -o main1 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main1 log.txt

Demo 1: Output redirection

        index    value   size    location
        1:       100     4       0xd7d494
        2:       100     4       0xd7d498
        3:       100     4       0xd7d49c
        4:       100     4       0xd7d4a0
        5:       100     4       0xd7d4a4
        6:       100     4       0xd7d4a8
        7:       100     4       0xd7d4ac
        8:       100     4       0xd7d4b0
        9:       100     4       0xd7d4b4
        10:      100     4       0xd7d4b8
        11:      100     4       0xd7d4bc
        12:      100     4       0xd7d4c0
        13:      100     4       0xd7d4c4
        14:      100     4       0xd7d4c8
        15:      100     4       0xd7d4cc

eric@Eric:~/Desktop/CS296-41$ cat log.txt
You successfully allocated an 1D array (15 elements of 4 bytes) at 0x18e1494
You freed an array of size 60 at 0x18e1494
eric@Eric:~/Desktop/CS296-41$
```

# Support 2: Array visualization

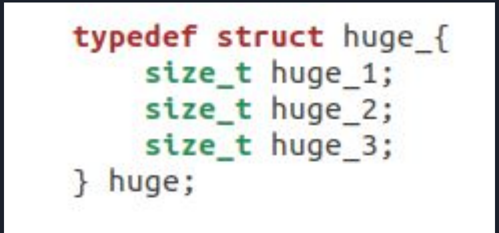
```
detail(arr, NULL, NULL, "Demo 2: Array visualization");
```



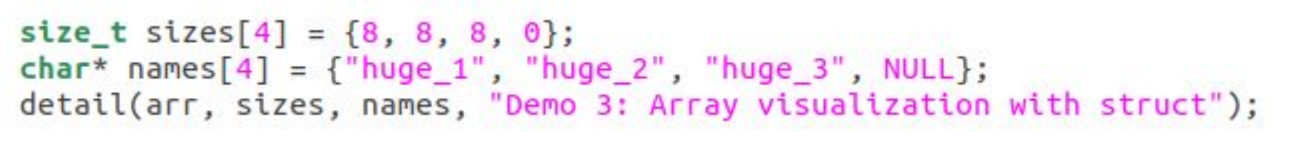
eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/CS296-41$ make demo2
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo2.c -o main2 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main2 log.txt

Demo 2: Array visualization

        index    value   size    location
        1:       100     4       0x121b494
        2:       100     4       0x121b498
        3:       100     4       0x121b49c
        4:       100     4       0x121b4a0
        5:       100     4       0x121b4a4
        6:       100     4       0x121b4a8
        7:       100     4       0x121b4ac
        8:       100     4       0x121b4b0
        9:       100     4       0x121b4b4
        10:      100     4       0x121b4b8
        11:      100     4       0x121b4bc
        12:      100     4       0x121b4c0
        13:      100     4       0x121b4c4
        14:      100     4       0x121b4c8
        15:      100     4       0x121b4cc

eric@Eric:~/Desktop/CS296-41$ 
```

# Support 3: Array visualization (with struct)

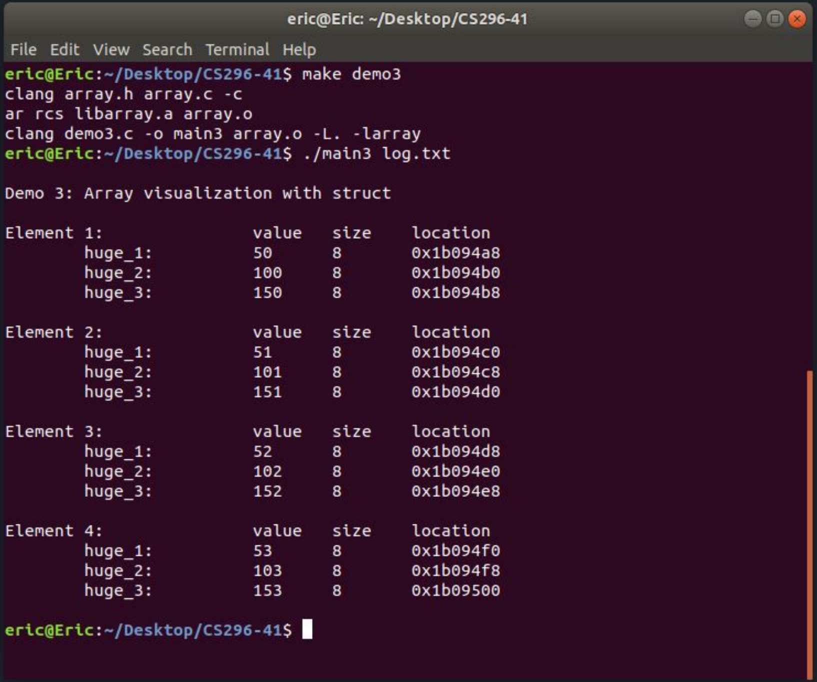


```c
typedef struct huge_{
    size_t huge_1;
    size_t huge_2;
    size_t huge_3;
} huge;
```



```c
size_t sizes[4] = {8, 8, 8, 0};
char* names[4] = {"huge_1", "huge_2", "huge_3", NULL};
detail(arr, sizes, names, "Demo 3: Array visualization with struct");
```

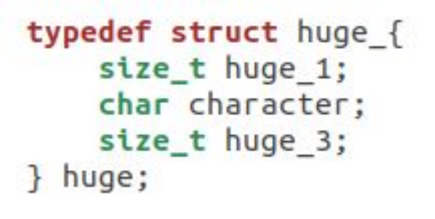
# Support 3: Array visualization (with struct)



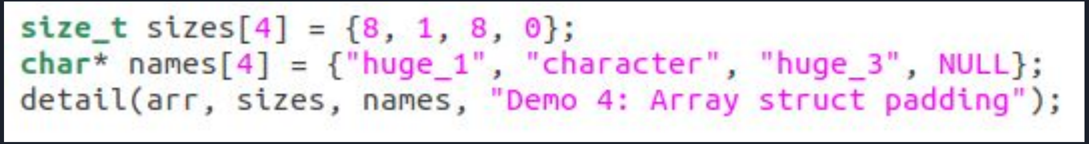
eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/CS296-41$ make demo3
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo3.c -o main3 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main3 log.txt

Demo 3: Array visualization with struct

Element 1:              value   size    location
        huge_1:         50      8       0x1b094a8
        huge_2:         100     8       0x1b094b0
        huge_3:         150     8       0x1b094b8

Element 2:              value   size    location
        huge_1:         51      8       0x1b094c0
        huge_2:         101     8       0x1b094c8
        huge_3:         151     8       0x1b094d0

Element 3:              value   size    location
        huge_1:         52      8       0x1b094d8
        huge_2:         102     8       0x1b094e0
        huge_3:         152     8       0x1b094e8

Element 4:              value   size    location
        huge_1:         53      8       0x1b094f0
        huge_2:         103     8       0x1b094f8
        huge_3:         153     8       0x1b09500

eric@Eric:~/Desktop/CS296-41$ 
```

# Support 4: Array struct padding

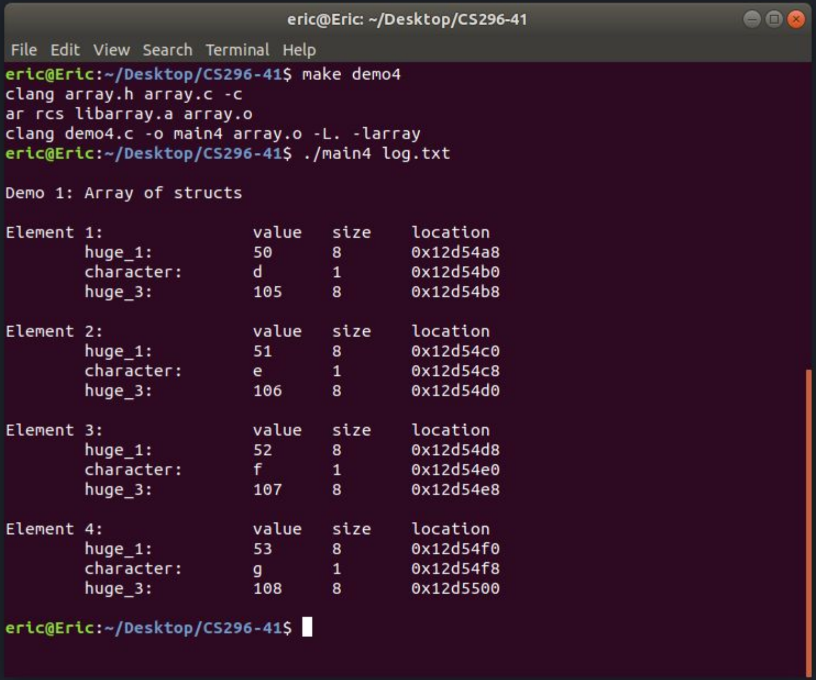


```c
typedef struct huge_{
    size_t huge_1;
    char character;
    size_t huge_3;
} huge;
```



```c
size_t sizes[4] = {8, 1, 8, 0};
char* names[4] = {"huge_1", "character", "huge_3", NULL};
detail(arr, sizes, names, "Demo 4: Array struct padding");
```

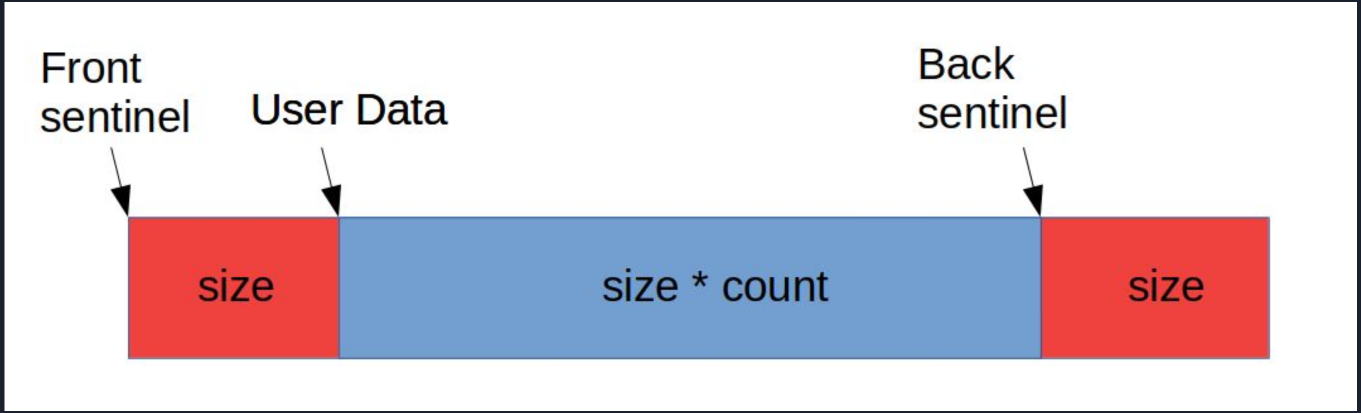
# Support 4: Array struct padding



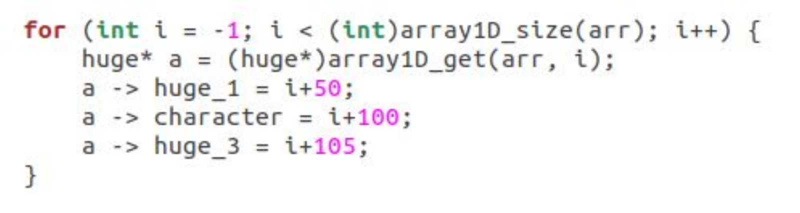
eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
eric@Eric:~/Desktop/CS296-41$ make demo4
clang array.h array.c -c
ar rcs libarray.a array.o
clang demo4.c -o main4 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main4 log.txt

Demo 1: Array of structs

Element 1:              value   size    location
        huge_1:         50      8       0x12d54a8
        character:      d       1       0x12d54b0
        huge_3:         105     8       0x12d54b8

Element 2:              value   size    location
        huge_1:         51      8       0x12d54c0
        character:      e       1       0x12d54c8
        huge_3:         106     8       0x12d54d0

Element 3:              value   size    location
        huge_1:         52      8       0x12d54d8
        character:      f       1       0x12d54e0
        huge_3:         107     8       0x12d54e8

Element 4:              value   size    location
        huge_1:         53      8       0x12d54f0
        character:      g       1       0x12d54f8
        huge_3:         108     8       0x12d5500

eric@Eric:~/Desktop/CS296-41$ 
```

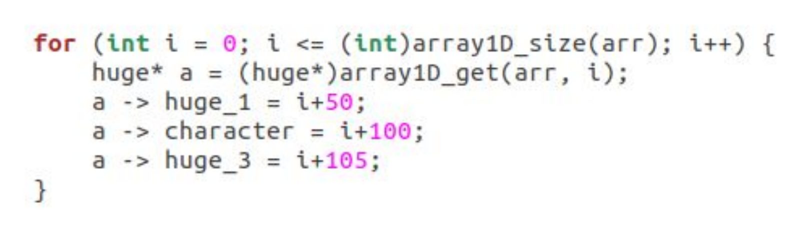
# Support 5: Array memory corruption check



# Support 5: Array memory corruption check

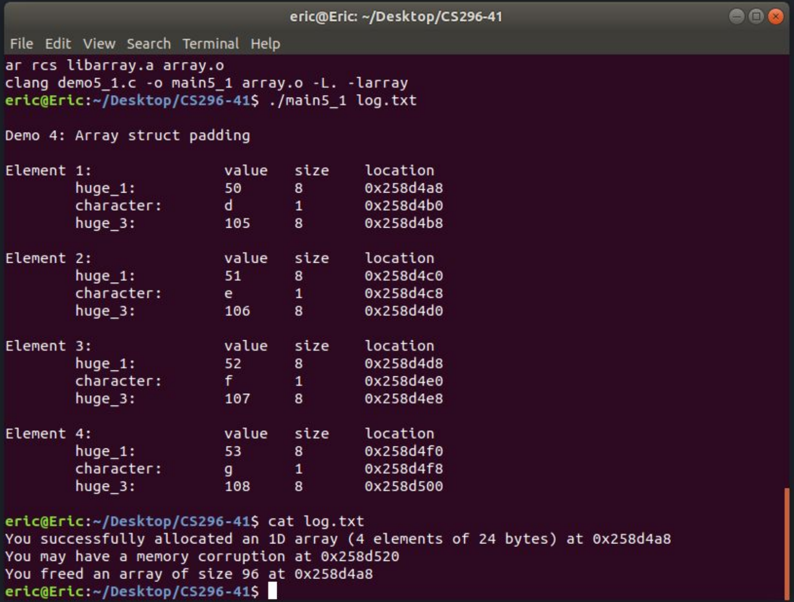


```c
for (int i = -1; i < (int)array1D_size(arr); i++) {
    huge* a = (huge*)array1D_get(arr, i);
    a -> huge_1 = i+50;
    a -> character = i+100;
    a -> huge_3 = i+105;
}
```

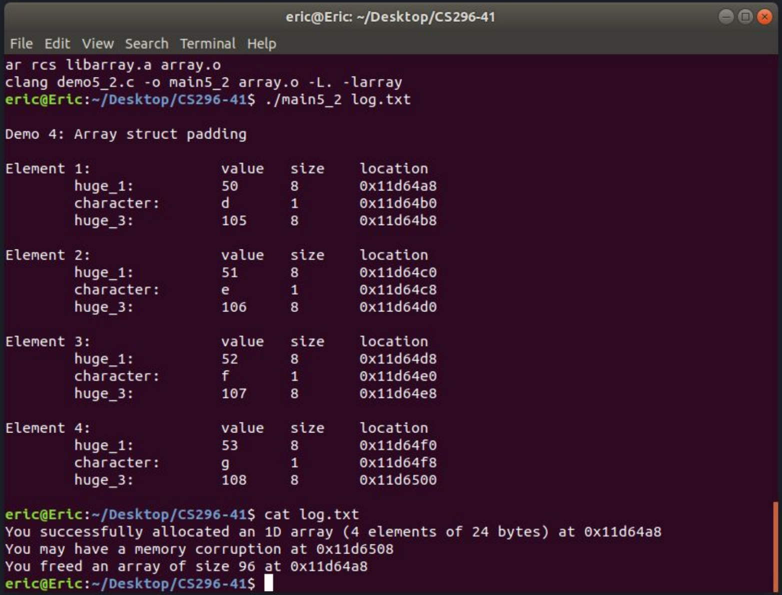


```c
for (int i = 0; i <= (int)array1D_size(arr); i++) {
    huge* a = (huge*)array1D_get(arr, i);
    a -> huge_1 = i+50;
    a -> character = i+100;
    a -> huge_3 = i+105;
}
```

# Support 5: Array memory corruption check


eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
ar rcs libarray.a array.o
clang demo5_1.c -o main5_1 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main5_1 log.txt

Demo 4: Array struct padding

Element 1:              value   size    location
        huge_1:         50      8       0x258d4a8
        character:      d       1       0x258d4b0
        huge_3:         105     8       0x258d4b8

Element 2:              value   size    location
        huge_1:         51      8       0x258d4c0
        character:      e       1       0x258d4c8
        huge_3:         106     8       0x258d4d0

Element 3:              value   size    location
        huge_1:         52      8       0x258d4d8
        character:      f       1       0x258d4e0
        huge_3:         107     8       0x258d4e8

Element 4:              value   size    location
        huge_1:         53      8       0x258d4f0
        character:      g       1       0x258d4f8
        huge_3:         108     8       0x258d500

eric@Eric:~/Desktop/CS296-41$ cat log.txt
You successfully allocated an 1D array (4 elements of 24 bytes) at 0x258d4a8
You may have a memory corruption at 0x258d520
You freed an array of size 96 at 0x258d4a8
eric@Eric:~/Desktop/CS296-41$
```


eric@Eric: ~/Desktop/CS296-41
File Edit View Search Terminal Help

```
ar rcs libarray.a array.o
clang demo5_2.c -o main5_2 array.o -L. -larray
eric@Eric:~/Desktop/CS296-41$ ./main5_2 log.txt

Demo 4: Array struct padding

Element 1:              value   size    location
        huge_1:         50      8       0x11d64a8
        character:      d       1       0x11d64b0
        huge_3:         105     8       0x11d64b8

Element 2:              value   size    location
        huge_1:         51      8       0x11d64c0
        character:      e       1       0x11d64c8
        huge_3:         106     8       0x11d64d0

Element 3:              value   size    location
        huge_1:         52      8       0x11d64d8
        character:      f       1       0x11d64e0
        huge_3:         107     8       0x11d64e8

Element 4:              value   size    location
        huge_1:         53      8       0x11d64f0
        character:      g       1       0x11d64f8
        huge_3:         108     8       0x11d6500

eric@Eric:~/Desktop/CS296-41$ cat log.txt
You successfully allocated an 1D array (4 elements of 24 bytes) at 0x11d64a8
You may have a memory corruption at 0x11d6508
You freed an array of size 96 at 0x11d64a8
eric@Eric:~/Desktop/CS296-41$
```

# Array limitations

1. Visualization for struct within structs
    - Expected limitation, partially solved by print(char* name, size_t* size, void* location) in header file
2. Memory corruption passing boundary tags
    - Also an expected limitation, cannot check for all memory corruptions

# Thank you

- Ophir and Steven
- And all the other TAs / CAs :)

# Questions ?