

Nursulu Musa

1. #include <stdio.h>

#include <stdlib.h>

#define PRINT_DELIMITER() printf("-----\n");

#define PRINT_MEM_SEG(seg, addr) printf("[%s]: %p\n", seg, addr);

static void func(void)

{

printf("hello\n");

}

int main(int argc, char *argv[])

{

PRINT_DELIMITER();

PRINT_MEM_SEG("CMD LINE ARGS", &argv[0]);

int stack_var = 25;

PRINT_DELIMITER();

PRINT_MEM_SEG("STACK", &stack_var);

int *heap_ptr = malloc(sizeof(int));

PRINT_DELIMITER();

PRINT_MEM_SEG("HEAP", heap_ptr);

static int data_var = 10;

PRINT_DELIMITER();

PRINT_MEM_SEG("DATA", &data_var);

```

PRINT_DELIMITER();

PRINT_MEM_SEG("TEXT", func);

return 0;
}

```

The screenshot shows a code editor with a file named `main.c`. The code defines a `main` function that prints memory segment addresses for various memory segments. The console output at the bottom shows the addresses for `[DATA]` and `[TEXT]`.

```

main.c
12 int main(int argc, char *argv[])
13 {
14     PRINT_DELIMITER();
15     PRINT_MEM_SEG("CMD LINE ARGS", &argv[0]);
16
17     int stack_var = 25;
18     PRINT_DELIMITER();
19     PRINT_MEM_SEG("STACK", &stack_var);
20
21     int *heap_ptr = malloc(sizeof(int));
22     PRINT_DELIMITER();
23     PRINT_MEM_SEG("HEAP", heap_ptr);
24
25     static int data_var = 10;
26     PRINT_DELIMITER();
27     PRINT_MEM_SEG("DATA", &data_var);
28
29     PRINT_DELIMITER();
30     PRINT_MEM_SEG("TEXT", func);
31
32     return 0;
33 }
34
input
[DATA]: 0x5939aaad4010
-----
[TEXT]: 0x5939aaad11a9

```

2. `#include <unistd.h>`

`#include <stdio.h>`

```

int main(int argc, char *argv[]){
    int *currentBreak = sbrk(0);
    printf("%p\n", currentBreak);
}

```

}

```
main.c
1  #include <unistd.h>
2  #include <stdio.h>
3
4  int main(int argc, char *argv[]){
5      int *currentBreak = sbrk(0);
6      printf("%p\n", currentBreak);
7  }
8
```

0x651ed34f3000

input

...Program finished with exit code 0
Press ENTER to exit console.

3. #include <unistd.h>

#include <stdio.h>

int main(int argc, char *argv[]){

void *currentBreak = sbrk(0x5);

printf("First increment of 0x5: %p\n", currentBreak);

currentBreak = sbrk(0x5);

printf("Second increment of 0x5: %p\n", currentBreak);

currentBreak = sbrk(0x5);

printf("Third increment of 0x5: %p\n", currentBreak);

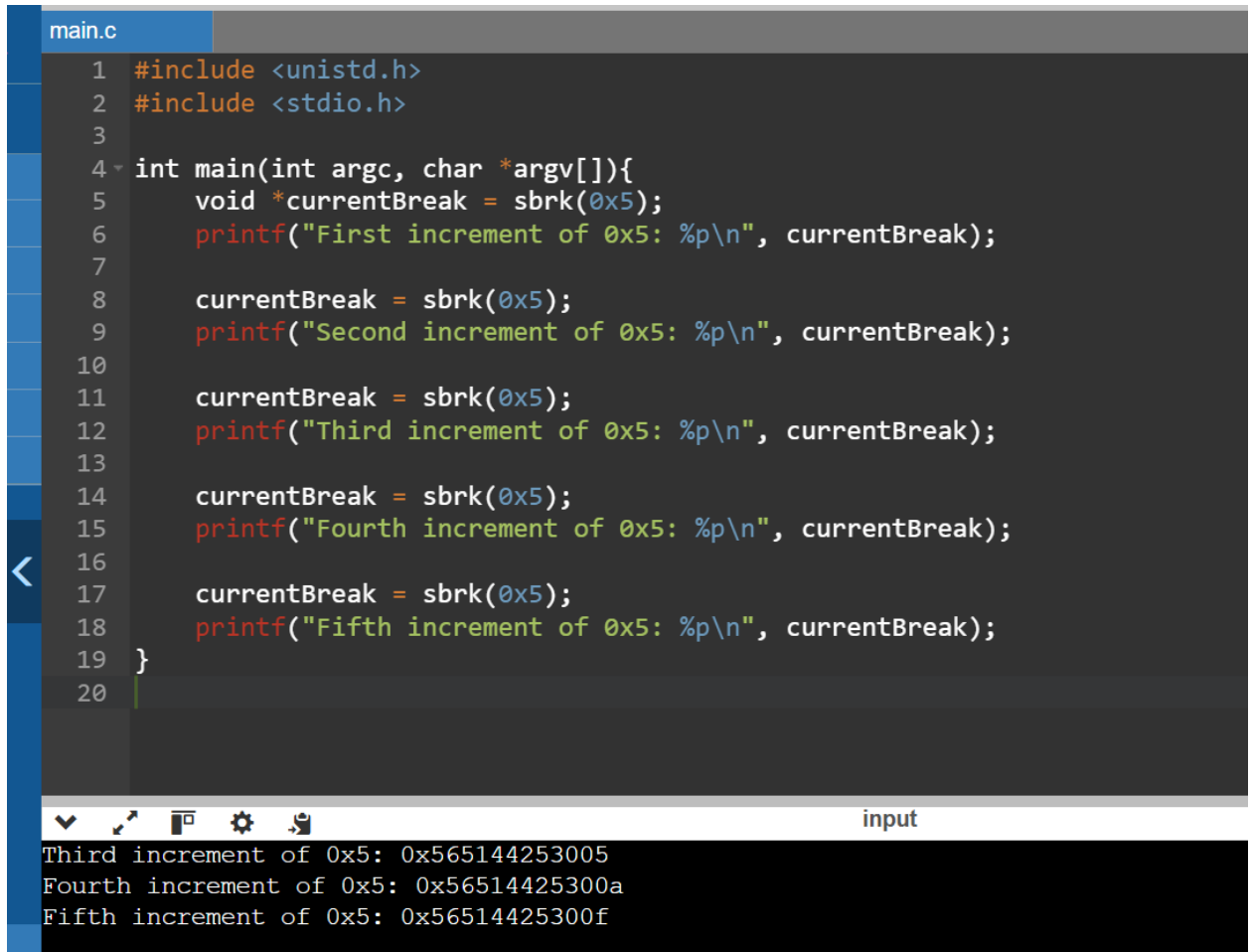
currentBreak = sbrk(0x5);

```
printf("Fourth increment of 0x5: %p\n", currentBreak);
```

```
currentBreak = sbrk(0x5);
```

```
printf("Fifth increment of 0x5: %p\n", currentBreak);
```

```
}
```



```
main.c
1  #include <unistd.h>
2  #include <stdio.h>
3
4  int main(int argc, char *argv[]){
5      void *currentBreak = sbrk(0x5);
6      printf("First increment of 0x5: %p\n", currentBreak);
7
8      currentBreak = sbrk(0x5);
9      printf("Second increment of 0x5: %p\n", currentBreak);
10
11     currentBreak = sbrk(0x5);
12     printf("Third increment of 0x5: %p\n", currentBreak);
13
14     currentBreak = sbrk(0x5);
15     printf("Fourth increment of 0x5: %p\n", currentBreak);
16
17     currentBreak = sbrk(0x5);
18     printf("Fifth increment of 0x5: %p\n", currentBreak);
19 }
20
```

input

```
Third increment of 0x5: 0x565144253005
Fourth increment of 0x5: 0x56514425300a
Fifth increment of 0x5: 0x56514425300f
```