

Week 6- Lecture 3

Dynamic Memory Allocation

Edited by: Dr. Saeid Pourroostaei Ardakani
Autumn 2021





Quiz!

Which one is False?

- A) Strtod converts a string to double
- B) sscanf reads using a standard keyboard
- C) Strcpy suffers from buffer overrun.
- D) sprintf writes to a string instead of screen.



Quiz!

Which one is False?

- A) Strtod converts a string to double
- B) sscanf reads using a standard keyboard**
- C) Strcpy suffers from buffer overrun.
- D) sprintf writes to a string instead of screen.



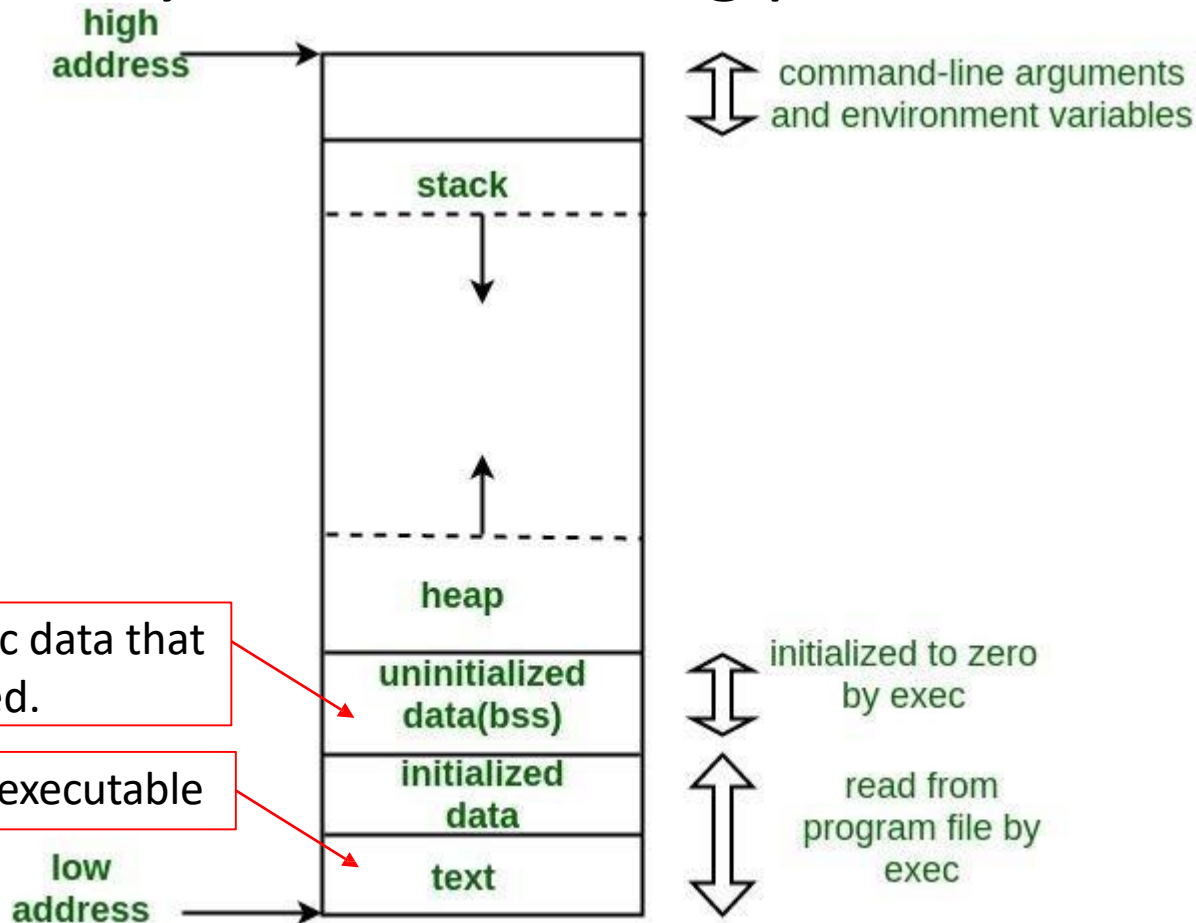
Overview

- Heap and Stack
- malloc and free



Memory Layout of C Programs

- Typical layout of a running process ...



Source: <https://www.geeksforgeeks.org/memory-layout-of-c-program/>




Memory Layout of C Programs (2)

- Note the size of the uninitialised data (bss).


```
2  #include <stdio.h>
3
4  int main(void)
5  {
6      return 0;
7  }
```

```
C:\Users\z2017233\Desktop>size dynamic.exe
   text    data     bss     dec     hex filename
   14212    1532      128    15872    3e00 dynamic.exe
```




```
10 #include <stdio.h>
11
12 int global;
13
14 int main(void)
15 {
16     return 0;
17 }
```

```
C:\Users\z2017233\Desktop>size dynamic.exe
   text    data     bss     dec     hex filename
   14212    1532      132    15876    3e04 dynamic.exe
```



```
20 #include <stdio.h>
21
22 int global;
23
24 int main(void)
25 {
26     static int i;
27     return 0;
28 }
```

```
C:\Users\z2017233\Desktop>size dynamic.exe
   text    data     bss     dec     hex filename
   14212    1532      136    15880    3e08 dynamic.exe
```




Memory Layout of C Programs (3)

- Note the size of the initialised data.


```
20 #include <stdio.h>
21
22 int global;
23
24 int main(void)
25 {
26     static int i;
27     return 0;
28 }
```

```
C:\Users\z2017233\Desktop>size dynamic.exe
   text    data     bss     dec     hex filename
  14212    1532     136    15880    3e08 dynamic.exe
C:\Users\z2017233\Desktop>
```



```
31 #include <stdio.h>
32
33 int global;
34
35 int main(void)
36 {
37     static int i = 100;
38     return 0;
39 }
```

```
C:\Users\z2017233\Desktop>size dynamic.exe
   text    data     bss     dec     hex filename
  14212    1536     132    15880    3e08 dynamic.exe
C:\Users\z2017233\Desktop>
```



Remember this!?

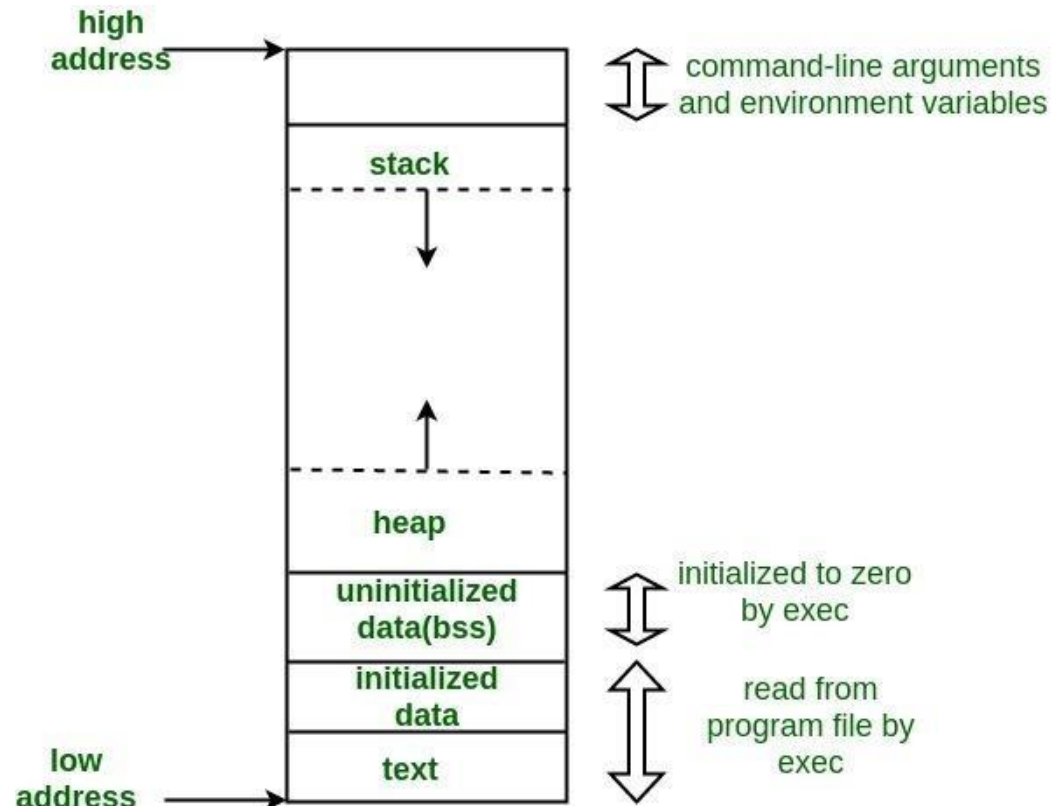
- The compiler allocates memory (i.e. stack) to store the function's parameters and the variables when the function is called.
- Once it's terminated, the memory is automatically deallocated.
- ... and **YES**, main is a function!!



Memory Layout of C Programs (5)

```
C:\Users\z2017233\Desktop>dynamic
0060FF2C
00407020
00407074
00404004
00401460
C:\Users\z2017233\Desktop>
```

```
53  #include <stdio.h>
54
55  int global;
56
57  int main(void)
58  {
59      static int i = 100;
60      static int j;
61
62      int k;
63
64      printf("%p\n", &k);
65      printf("%p\n", &j);
66      printf("%p\n", &global);
67      printf("%p\n", &i);
68      printf("%p\n", main);
69
70
71      return 0;
72  }
```



Source: <https://www.geeksforgeeks.org/memory-layout-of-c-program/>



Overview

- Heap and Stack
- **malloc and free**



Heap (Unlike Stack ...)

- The segment where dynamic memory allocation usually takes place.
- Memory doesn't get deallocated at the end of a function call.
- Manage by the programmer using e.g. malloc, and free.

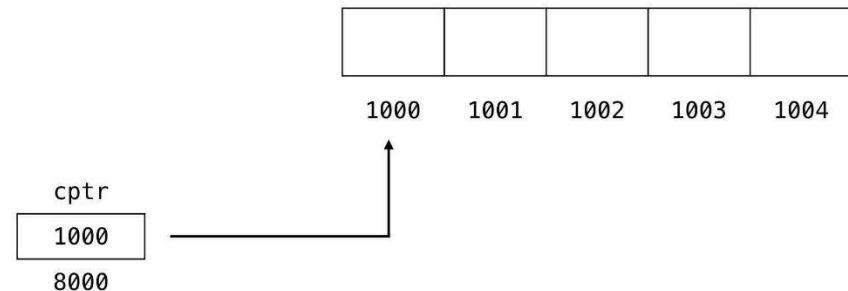
```
75  #include <stdio.h>
76  #include <stdlib.h>
77
78  int global;
79
80  int main(void)
81  {
82      static int i = 100;
83      static int j;
84
85      int k;
86
87      int *p = malloc(sizeof(int));
88
89      printf("%p\n", &k);
90      printf("%p\n", &p);
91      printf("%p\n", &global);
92      printf("%p\n", &j);
93      printf("%p\n", &i);
94      printf("%p\n", main);
95
96      free(p);
97
98
99      return 0;
100 }
```

Dynamic Memory Allocation

- Create dynamic data structures that can change size e.g., lists, trees, graphs.

 CLASSROOM

```
char *cptr = (char *) malloc (5 * sizeof(char));
```



Source: <https://www.dyclassroom.com/c/c-dynamic-memory-allocation-malloc-function>



malloc

- Returns a pointer to a newly allocated block of memory in the heap.
- Size is determined in bytes.
- Use

```
int *p = malloc(sizeof(int));  
char *q = malloc(sizeof(char));
```



free

- To deallocate the block of memory after you have finished using.
- Trying to free memory not allocated by malloc is an error.
- Trying to free the same memory multiple times is an error.
- `free(p);`



free (2)

- If forget to free memory which no longer required, it can make your program use more and more memory the longer it is running.
- When the program exits, the OS will reclaim all of the memory, even if it has not been freed.





Example: Reusable Prompt

- To print a prompt then read in a string.

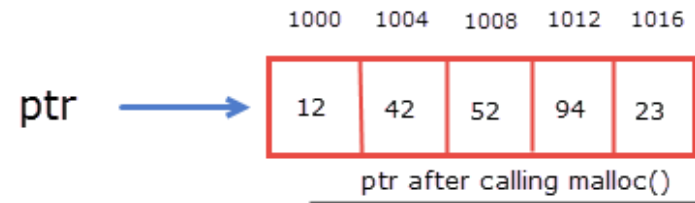
```
103  #include <stdio.h>
104  #include <stdlib.h>
105
106  char *prompt(const char *mesg, const int limit);
107
108  int main(int argc, char *argv[])
109  {
110      char *name = prompt("Who are you?\n", 20);
111      if(name == NULL)
112      {
113          printf("Error\n");
114      }
115      else
116      {
117          printf("Hello %s!\n", name);
118          free(name);
119      }
120
121      return 0;
122  }
123
124  char *prompt(const char *mesg, const int limit)
125  {
126      char *name;
127      name = malloc(sizeof(char) * (limit + 1));
128      if(name == NULL)
129      {
130          return NULL;
131      }
132
133      printf("%s", mesg);
134      scanf("%s", name);
135      return name;
136  }
```



realloc

- To resize the previously allocated memory.

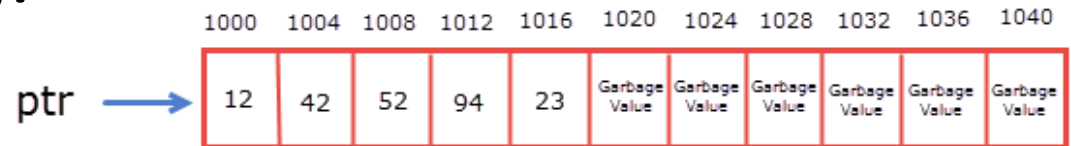
```
p = (int*)malloc(5*sizeof(int));
```



```
p = (int*)realloc(p, 11*sizeof(int));
```

Now two conditions may arise:

1st case: If sufficient memory is available after address 1016, then the address of ptr doesn't change.



2nd case: If sufficient memory is not available after address 1016, then the realloc() function allocates memory somewhere else in the heap and copies the all content from old memory block to the new memory block. In this case the address of ptr changes.



Source: <https://overiq.com/c-programming-101/the-realloc-function-in-c/>



Example: realloc

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

```
int main () {
    char *str;
```

```
    str = (char *) malloc(sizeof(char)*15);
    strcpy(str, "tutorialspoint");
    printf("String = %s, Address = %p\n", str, str);
```

```
    str = (char *) realloc(str, 25*sizeof(char));
    strcat(str, ".com");
    printf("String = %s, Address = %p\n", str, str);
```

```
    free(str);
    return(0);
}
```

Output:

String = tutorialspoint, Address = 0xd204010

String = tutorialspoint.com, Address = 0xd204010



Summary

- Heap and Stack
- malloc and free



Quiz!

Which one is False?

- A) Compiler allocates memory (i.e. stack) to store the function's parameters and the variables when the function is called.
- B) Stack is a linear data structure, whereas heap is a hierarchical data structure.
- C) Heap addresses dynamic memory allocation.
- D) Stack is random order, whereas heap contains contiguous blocks.

Quiz!

Which one is False?

- A) Compiler allocates memory (i.e. stack) to store the function's parameters and the variables when the function is called.
- B) Stack is a linear data structure, whereas heap is a hierarchical data structure.
- C) Heap addresses dynamic memory allocation.
- D) Stack is random order, whereas heap contains contiguous blocks.**



Quiz!

Which one is False?

- A) Heap memory doesn't get deallocated at the end of a function call.
- B) Dynamic memory allocation is supported during run time.
- C) free the same memory multiple times is fine.
- D) Malloc returns a pointer to a newly allocated block of memory in the heap.



Quiz!

Which one is False?

- A) Heap memory doesn't get deallocated at the end of a function call.
- B) Dynamic memory allocation is supported during run time.
- C) free the same memory multiple times is fine.**
- D) Malloc returns a pointer to a newly allocated block of memory in the heap.

