Pointers

Dynamic Memory Allocation

Introduction

- Whenever you declare a variable in C, you are effectively reserving one or more locations in the computer's memory to contain values that will be stored in that variable. This is compile time allocation. For example,

  ```
  int my_array[10];
  int n = 20;
  ```

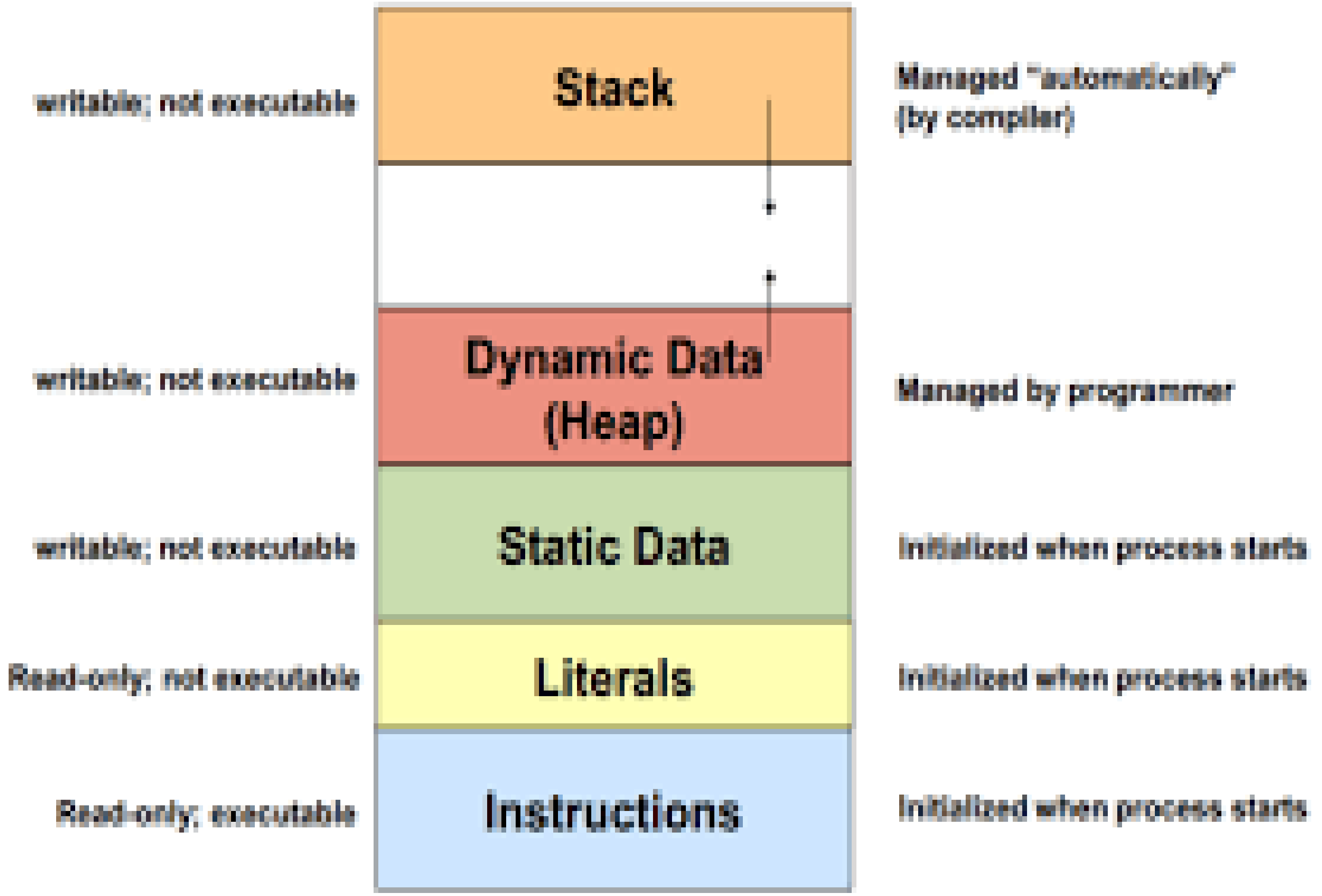
- In some cases, the size of the variable may not be known until run-time, such as in the case of where the user enters how many data items will be entered, or when data from a file of unknown size will be read in.

- For dynamic memory allocation, pointers are crucial

Stack vs Heap Memory Allocation

- Memory in a C program can either be allocated on stack or heap.
- Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM .
- In static memory allocation, the variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is deallocated. This all happens using some predefined routines in compiler. Programmer does not have to worry about memory allocation and deallocation of stack variables.
- Variables allocated on the heap have their memory allocated at run time and the programmer has to deallocate it.
- You can use the stack if you know exactly how much data you need to allocate before compile time and it is not too big. You can use heap if you don't know exactly how much data you will need at runtime or if you need to allocate a lot of data.

Simplified Memory Layout in a Linux System



Variable-length Arrays

- The size of Variable-length Arrays is established at run time. For example,

```
int n;
printf( “Enter the array size: ");
scanf( "%d", &n );
int a[n];
```

- Contrary to what their name implies, you cannot change the size of a variable-length array after it has been defined. "Variable-length" simply means that the size isn't fixed at compile time.

- Since their size isn't set until runtime, variable-length arrays may not be declared at file scope or with the static keyword, nor can they be declared with an initializer list.

- Exactly how the space for VLAs is managed is up to the implementation; it usually is taken from the stack.

Dynamic Memory Allocation

- Dynamic allocation requires two steps:
  - Creating the dynamic space.
  - Storing its **address** in a **pointer** (so that the space can be accessed)

- De-allocation:
  - Deallocation is the "clean-up" of space being used for variables or other data storage

Library functions

- To allocate and deallocate memory dynamically, the following library functions are used.

  - void *malloc(size_t size);
  - void *calloc(size_t nItems, size_t size);
  - void *realloc(void *ptr, size_t size);
  - void free(void *ptr);

- These functions are defined in the <**stdlib.h**> header file.

Use sizeof() with memory allocation functions

- Assuming size of objects can be misleading & is bad style, so use `sizeof(type).` Using sizeof operator makes code readable and portable.
- Many years ago an `int` was 16 bits, and programs assumed it was 2 bytes

Memory Leaks

- A memory leak occurs when you have dynamically allocated memory, using malloc() or calloc() that you do not free properly.
- As a result, this memory is lost and can never be freed, and thus a memory leak occurs.
- It is vital that memory leaks are plugged because they can cause system-wide performance issues as one program begins to hog all the memory, affecting access to the resources for other programs.

Homework

1. What is a segmentation fault?
2. What is the difference between variable length array and dynamic memory allocation?
3. Explain dangling pointer, NULL pointer, void pointer and wild pointer?