18ES611 Embedded System Programming

Sarath tv

Memory layout

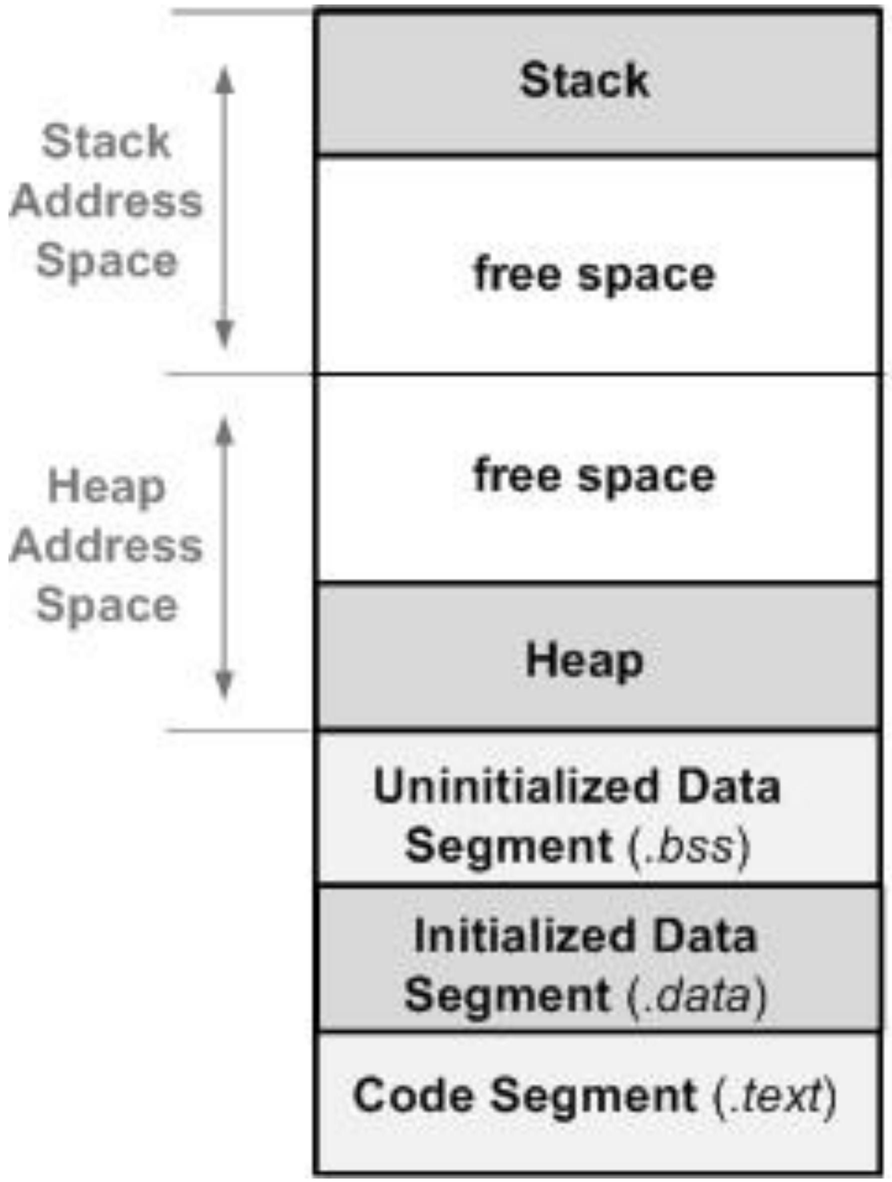
Where in memory are my variables stored in C?

Program code and data stored in the same memory space.

Text and **data segments**.

Modern systems use a **single text segment**, **more than one segment** for **data**, depending upon the storage class of the data being stored there.

1. Text or code segment
2. Initialized data segments
3. Uninitialized data segments
4. Stack segment
5. Heap segment


Stack
Stack Address Space
free space
free space
Heap Address Space
Heap
Uninitialized Data Segment (.bss)
Initialized Data Segment (.data)
Code Segment (.text)

Code Segment (.text)

This segment stores the executable program code (the machine instructions).

Variables defined with the *const* type qualifier can also be placed in the code segment.

It is a **read-only segment** stored in **a non-volatile memory**.

The code segment does not contain program variables like local variable (also called as automatic variables in C), global variables, etc.

the code segment can also contain **read-only string literals**.

For example, **`printf("Hello, world")`** then string "Hello, world" gets created in the code/text segment.

Data Segments

Data segment stores program data.

This data could be in form of **initialized** or **uninitialized variables**, and it could be **local** or **global**.

Four sub-data segments (initialized data segment, uninitialized or .Bss data segment, stack, and heap)

- ❖Initialized data segments
- ❖Uninitialized data segments
- ❖Stack segment
- ❖Heap segment

Uninitialized data segment

This segment is also known as bss.

Contains:

- ❑ Uninitialized global variables
- ❑ Uninitialized constant global variables.
- ❑ Uninitialized local static variables.

Any **global** or **static local variable** which is **not initialized** will be stored in the uninitialized data segment

If you declare a **global variable** and **initialize** it as **0** or **NULL** then still it would go to uninitialized data segment or bss.

This segment stores all global and local variables that are initialized to zero or do not have explicit initialization in the source code.

The memory locations of **variables** stored in this segment are **initialized** to **zero before** the **execution** of the **program starts**.

Uninitialized data segment, often called the “**bss**” segment, named after an ancient assembler operator that stood for “**block started by symbol**.”

static int i;

int j; (a global variable)

Initialized Data or Data Segment

Initialized data segment, usually called simply the data segment.

Contains the **global variables** and **static variables** that are **initialized** by the **programmer**.

Not read-only.

This segment can be further classified into initialized read-only area and initialized read-write area.

Global string **`char s[] = "hello world"`** ,

`Int debug=1` (global)

Would be stored in initialized read-write area.

Global c statement like `const char* string = "hello world"`

"hello world" to be stored in initialized read-only area and the character pointer variable string in initialized read-write area.

`Static int i = 10` will be stored in data segment and global int i = 10 will also be stored in data segment

- This segment stores all global variables (with storage class specifier *static* or *extern*) and local *static* variables that have defined initial values different than zero..
- Initialized data segment stores:
- Initialized global variables (including pointer variables)
- Initialized constant global variables.
- Initialized local static variables.
- For example: global variable `int globalVar = 1;` or static local variable `static int localStatic = 1;` will be stored in initialized data segment.
- The **size** of this **segment** is determined by the size of the values in the program's source code, and **does not** change at **run time**.

/*****Identify Memory location for these variable *****/

```
#include<stdio.h>
int variable_1=192;
 int variable_2;
int variable_3 = 0;

int main() {
static int variable_2=67;
/* ... Function code ... */

}
```

THANK YOU!!!!!