ESP_LEC_3

Memory Layout of C Program.

For a running program both the machine instructions (program code) and data are stored in the same memory space. The memory is logically divided into text and data segments. Modern systems use a single text segment to store program instructions, but more than one segment for data, depending upon the storage class of the data being stored there.

These segments can be described as follows:

1. Text or Code Segment
2. Initialized Data Segments
3. Uninitialized Data Segments
4. Stack Segment
5. Heap Segment

Text or Code Segment

Code segment, also known as *text segment* contains machine code of the compiled program. The text segment of an executable object file is often read-only segment that prevents a program from being accidentally modified.

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. Data segment is further divided into four sub-data segments (initialized data segment, uninitialized or .bss data segment, stack, and heap) to store variables depending upon if they are local or global, and initialized or uninitialized.

Initialized Data or Data Segment

Initialized data or simply *data segment* stores all global, static, constant, and external variables (declared with extern keyword) that are initialized beforehand.

Uninitialized Data or .bss Segment

uninitialized data or *.bss segment* stores all uninitialized global, static, and external variables (declared with extern keyword). Global, external, and static variable are by default initialized to zero. This section occupies no actual space in the object file; it is merely a place holder. Object file formats distinguish between initialized and uninitialized variables for space efficiency; uninitialized variables do not have to occupy any actual disk space in the object file.

Size operator to verify the memory layout