```
#include <stdio.h>
struct car
{
int size;
char *name;
};
int main()
{
struct car car1;

car1.name = "bmw";
car1.size = 10;

struct car car2;

car2.name = "ferrari";
car2.size = 20;

printf(" car1 %d, %s\n",car1.size,car1.name);
printf(" car2 %d, %s\n",car2.size,car2.name);
return 0;
}
```

in this code is struct car does not loading in memory until you make a variable of type struct car car1
I meen now var store in memory have the name size or name only car1.size and car1.name

please explain what happen when I init struct and make variable of his type in memory lyout

In C, a `struct` (short for "structure") is a way to group different types of variables into a single compound data type. When you define a struct, you are creating a new data type, but at this point, no memory is allocated yet. Memory is allocated only when you create an instance (variable) of that struct type.

Here's the breakdown of what happens with your `car` struct and its instances:

1. `struct car`: When you define `struct car { ... };`, you're telling the compiler about a new data type. This definition by itself does not cause any memory to be allocated. It's like a blueprint that says, "A `car` consists of an `int` named `size` and a `char` pointer named `name`."
2. `struct car car1;`: When you declare a variable `car1` of type `struct car`, memory is allocated on the stack for this instance of `car`. The amount of memory allocated is enough to store an `int` (for the `size` field) and a `char` pointer (for the `name` field). The exact amount of memory depends on the system and the compiler, as different systems may have different sizes for these types. However, typically an `int` might be 4 bytes and a pointer might be 4 bytes on a 32-bit system or 8 bytes on a 64-bit system.
3. Initializing `car1`: When you assign values to `car1.size` and `car1.name`, you're storing data into the memory that was allocated for `car1`. The `size` field gets the integer value `10`. The `name` field gets the address of the string literal `"bmw"`. Note that the string literal is stored in a read-only section of the memory (often called the "text segment" or "string literal pool"), and only the pointer to it is stored in the `car1` instance.
4. `struct car car2;`: Similarly, when you declare `car2`, another block of memory is allocated separately for `car2`, with space for its `size` and `name` fields. This memory is distinct from the memory for `car1`.
5. Initializing `car2`: Assigning values to `car2.size` and `car2.name` stores the integer `20` in `car2.size` and the address of the string literal `"ferrari"` in `car2.name`.

In memory, you have something like this

From ChatGPT, designed with FancyGPT