Pointers

September 28, 2019
Runtime Memory Layout

- **Text segment**: machine instructions of program
- **Data segment**: constants, global variables (static or not) and local static variables
- **Heap**: dynamically allocated space, `malloc()`
- **Stack**: runtime call stack
  - Actual parameters (arguments) for functions
  - Local (automatic) variables
  - Return address
  - Returned value
See demo.
C program’s stack segment is consisted of nested function frames or stack frames.

the main frame is the most outside frame, and the starting point of the program.

Each function call will start one new frame in the stack, and after exit of function, its stack frame gets destroyed with every thing in it.

Stack frames works like stack structure, the first created frame gets destroyed at last, which is the main, and the last created stack frame gets destroyed first. FILO.

Stack frame growing from high address to low address.
Local variable

- See demo.
See demo.

Are "array" and "&array" the same thing? See demo.
See demo.
The functions malloc and calloc obtain blocks of memory dynamically.

```c
void *malloc(size_t n)
```
returns a pointer to \( n \) bytes of uninitialized storage, or NULL if the request cannot be satisfied.

```c
void *calloc(size_t n, size_t size)
```
returns a pointer to enough free space for an array of \( n \) objects of the specified size, or NULL if the request cannot be satisfied. The storage is initialized to zero.

```c
free(p)
```
frees the space pointed to by \( p \), where \( p \) was originally obtained by a call to malloc or calloc.
In C, a function itself is not a variable, but it is possible to define pointers to functions, which can be assigned, placed in arrays, passed to functions, returned by functions, and so on.
Complicated Declarations

See demo.