

Lecture 9:

The Stack

CSE 29: Systems Programming and Software Tools

Aaron Schulman (Shalev)



Finishing up

Hash: An integer as a fingerprint for bytes



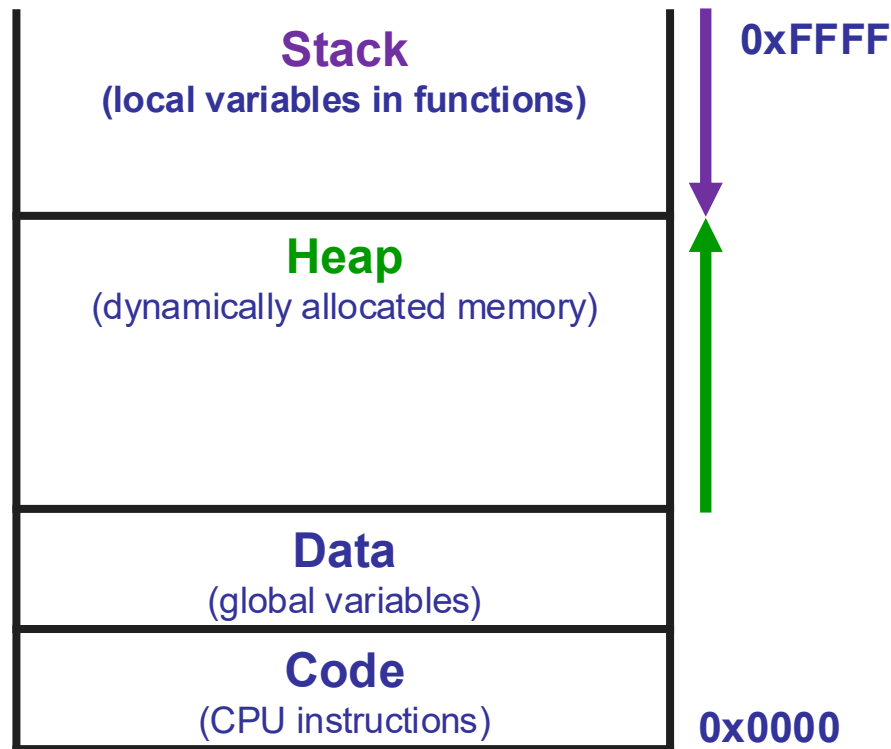
- Hash: A linear mathematical function that maps bytes to an integer
 - ...a really really big integer: 2^{256} (256-bits)
- Hashes are fingerprints that represent those bytes
 - » However, they are one-way functions
 - » The integer that makes up the fingerprint can not be reversed to find the bytes that made it
 - » This makes them particularly useful for use in password security
 - » They must *uniquely* represent those bytes or they are insecure
- There are many hash functions that you can come up with:
 - MD5 [Rivest] (1992) – 128-bits (broken in 2008)
 - SHA-1 [NSA] (1993) – 160-bits (broken in 2017)
 - SHA-256 [NSA] (2001) – 256-bits (still secure as far as we know)

The Stack – Memory for running functions

- When a program is running, each function that is called gets a segment of the memory to store values. This is called the function's stack frame.
- The stack stores “**local variables**” declared in a function (including params)
- It also stores the address of where in the code a function should return to*
- The stack grows to fit more values from the function if needed
- However, once a function is done executing, the stack frame in memory no longer belongs to the function and is able to *reused* for other functions
 - Because of this, local variables in functions get overwritten once the function is done
 - This is why we should NEVER return a pointer to a local variable on a function's stack frame because it may get overwritten

Where is the stack in memory?

- ❑ Grows down from high addrs to low
- ❑ Each process (running program) gets its own stack.
- ❑ If new values are added to the stack when a function is called, when does the stack grow a lot?



What happens when a function is done?

- ❑ Does a computer delete the data a function uses after its done?
 - ❑ No, there's no meaningful reason to "delete" the data. Some other code will overwrite it.
- ❑ Once memory is written, it stays that value until something else overwrites it.
 - ❑ This is one reason you always should initialize (e.g., zero out) variables and arrays before you use them!

Review Questions

1. 1. If you change one bit of binary data being hashed, how many bits of the hash will change?
2. 2. When you add variables to a function, will the function's stack frame contain lower or higher memory addresses than it did before?
3. 2. After a function is done, what happens to its local variables in memory?