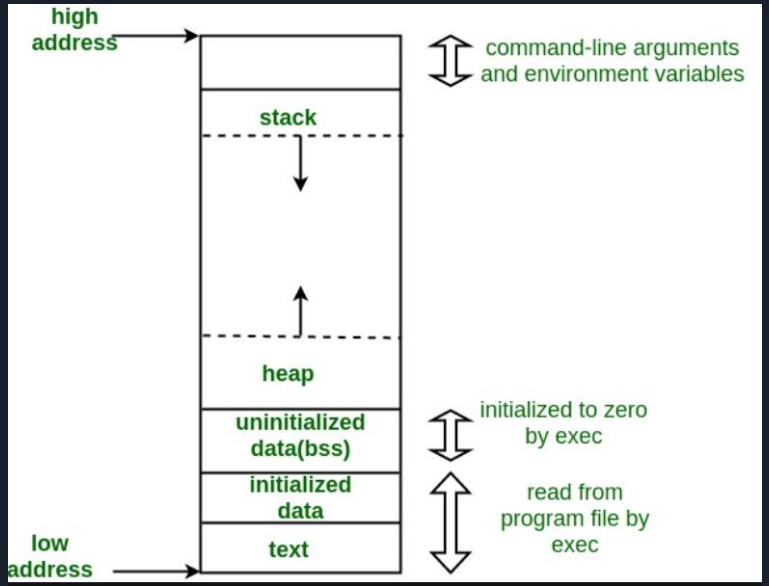
CS 24000 L04

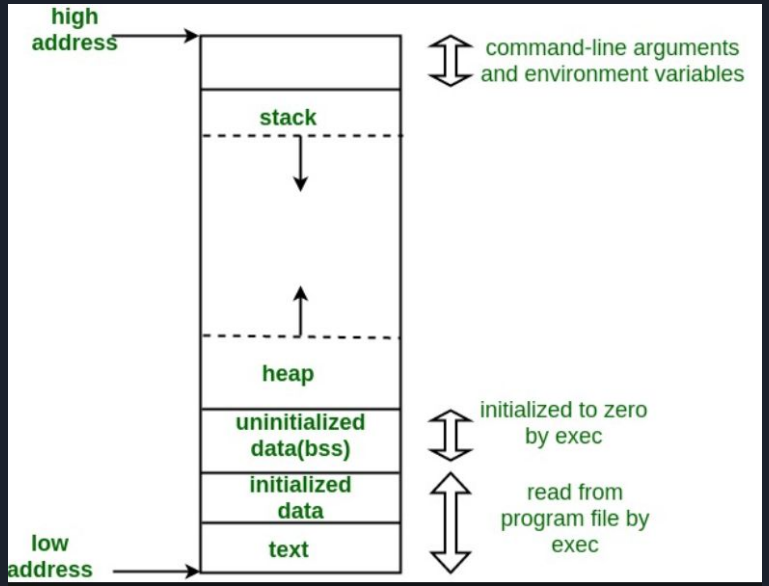
Week 7 - Memory layout, buffer overflows and the dangers of strcpy

Memory Layout



1. Text segment: Contains executable instructions
2. Initialized data: Global variables and static variables that are initialized by the programmer
3. Uninitialized data (bss): Statically allocated variables that are declared but have not been assigned a value yet

Memory Layout



4. Stack: Local variable storage
5. Heap: Dynamic memory allocation usually takes place; requires pointers to access it

Stack and heap grow in opposite direction and when the pointer meets, free memory is exhausted.

strcpy

The strcpy() function does not stop until it sees a zero (‘\0’) in the source string. If the source string is longer than the length assigned, strcpy() will overwrite some portion of the stack above the buffer. This is called a buffer overflow!

```
char buf[10] = “”;
char src[12] = “hello world”;
strcpy(buf, src);
```

h	e	l	l	o		w	o	r	l
d	\0	?	?	?	?	?	?	?	?

Possible seg fault/loss of data

Buffer Overflow vs. Stack Overflow

If a buffer overflow overwrites part of the stack frame, catastrophic effects can occur! If a buffer overflow overwrites part of memory outside of a function, it is called a stack overflow, and can be used as an attack to...

- Overwrite the return address of a function
- Overwrite the return value of a function
- Depending on the size of the buffer - Even overwrite the text section!

There are countermeasures to this, such as stack canaries and address space layout randomization (ASLR), but these are a bit advanced for this topic

Just know that copying too many bytes into a buffer can do some gnarly things

Use strncpy! (char *strncpy(char *dest, const char *src, size_t n)

Copies up to n characters from src to dest. In a case where the length of src is less than that of n, the remainder of dest will be padded with null bytes. Prevents buffer overflow.

Remember: Terminate dest by ‘\0’.