Data Segment and Linking

Source

Adapted from

- IA OS
- IB C and C++
    - `static`, `auto`, `extern`
- IB Compiler Construction
- IB Concurrent System
    - Threads
- Wiki-Data segment

Program memory

Code / Text segment

Program code and constants (string literals).

- read-only and fixed size.
- could be shared over all instances

```
char * message = "This is a string literal.";
```

Static Data

- Global variables
    - both `static`
        - preventing variables or function from being called externally
    - and `extern`
- Local `static` variables
    - retain its value between function calls

Initialized static data segment

```
(extern) int i = 3;
static int b = 2023;
void foo (void) {
    static int c = 2023;
}
```

Uninitialized static data segment / Block Starting Symbol

Both above variables and constants

- that do not have explicit initialization in source code.
- will be initialized to zero in C by `exec`

```
static int i;
static char a[12];
```

Heap

Dynamically allocated memory

- commonly begins at the end of the BSS segment and grows to larger addresses from there.
- `malloc`, `calloc`, `realloc`, and `free`
    - which may use the brk and sbrk system calls to adjust its size (mmap/munmap to reserve/unreserve potentially non-contiguous regions of virtual memory into the process' virtual address space).

```
ptr = (int*)malloc(n * sizeof(int));
free (ptr);
```

- The heap segment is shared by all threads, shared libraries, and dynamically loaded modules in a process.

Stack

- `auto` variables are also allocated on the stack.
    - function parameters, local variables

The call stack

- Typically located in the higher parts of memory.
    - LIFO structure
- stack frame
    - the set of values pushed for one function call
    - consist at minimum of a return address.
- stack pointer register
    - tracks the top of the stack
    - adjusted each time a value is "pushed" onto the stack

Note:

- The stack segment traditionally adjoined the heap segment and they grew towards each other
- when the stack pointer met the heap pointer, free memory was exhausted.

```
void f(int k){
    k++;
}

void main() {
    (auto) int j = 3;
    f(j);
}
```

Memory Layout

Addr	Data Segment	Include	Note
0x 8000	int argc, char *argv[]	command-line	
	Stack ↓	auto	
	Heap ↑	dynamic	
	Uninitialized static bss	variables const	initialized to 0 by exec
	Initialized static data	variables	
0x 0000	Code / Text	Program, const	

```
int argc, char *argv[]
```