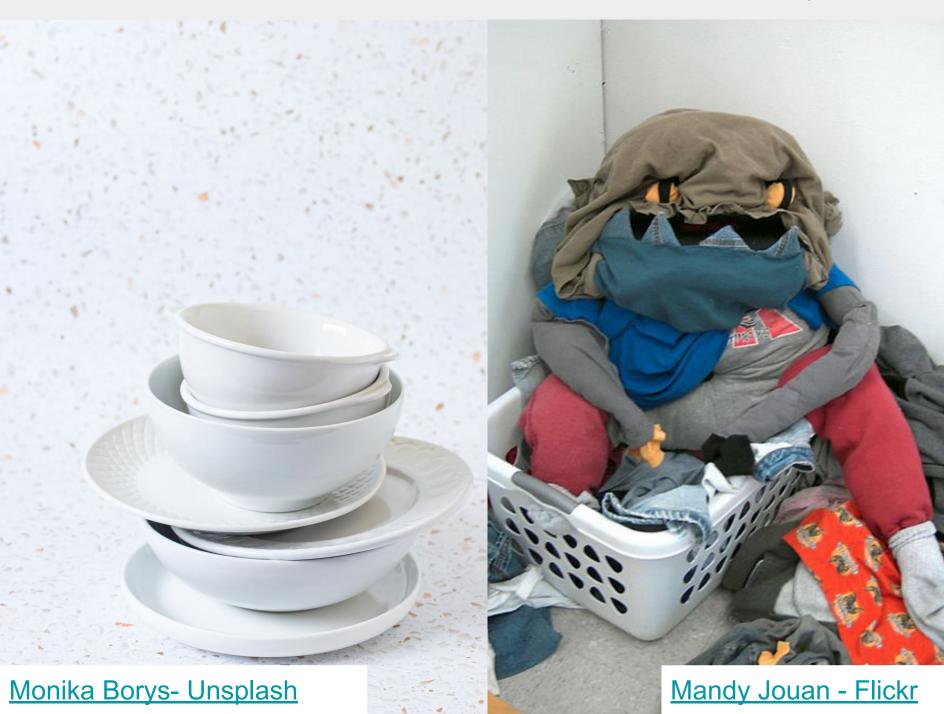
Stack vs Heap

Adapted from materials by Dr. Carrier



Two areas of memory

- Two areas of memory
- The stack is a stack data structure

- Two areas of memory
- The stack is a stack data structure
- The heap is not necessarily a heap

- Two areas of memory
- The stack is a stack data structure
- The heap is not necessarily a heap
- Variables are in one or the other
 - Which one depends on how variable is declared!

- Two areas of memory
- The stack is a stack data structure
- The heap is not necessarily a heap
- Variables are in one or the other
 - Which one depends on how variable is declared!
- So far, we've only dealt with the stack

New block (scope) for each function called

- New block (scope) for each function called
 - Function local variables are stored in block

- New block (scope) for each function called
 - Function local variables are stored in block
 - When function returns, that block is freed

- New block (scope) for each function called
 - Function local variables are stored in block
 - When function returns, that block is freed
 - Those variables are gone!

Dynamically allocated memory

- Dynamically allocated memory
 - Not tied to a particular functions scope

- Dynamically allocated memory
 - Not tied to a particular functions scope
 - Can be allocated and freed at any time

- Dynamically allocated memory
 - Not tied to a particular functions scope
 - Can be allocated and freed at any time
- We must free any memory we allocate

- Dynamically allocated memory
 - Not tied to a particular functions scope
 - Can be allocated and freed at any time
- We must free any memory we allocate
 - What happens if not?

- Dynamically allocated memory
 - Not tied to a particular functions scope
 - Can be allocated and freed at any time
- We must free any memory we allocate
 - What happens if not?
 - Memory leaks!

- Dynamically allocated memory
 - Not tied to a particular functions scope
 - Can be allocated and freed at any time
- We must free any memory we allocate
 - What happens if not?
 - Memory leaks!
 - You can lose access (pointer) to allocated memory
 - Thus your program can't free it
 - Usually cleaned up by OS when program exits

Stack allocates / deallocates automatically

- Stack allocates / deallocates automatically
 - Often limited in size

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)
- Heap allows flexibility, but require more thought

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)
- Heap allows flexibility, but require more thought
 - Larger arrays

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)
- Heap allows flexibility, but require more thought
 - Larger arrays
 - Arrays that persist after a function returns

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)
- Heap allows flexibility, but require more thought
 - Larger arrays
 - Arrays that persist after a function returns
 - Amount of memory is unknown at compile time

- Stack allocates / deallocates automatically
 - Often limited in size
 - All allocations on stack are static (sizes are set at compile time)
- Heap allows flexibility, but require more thought
 - Larger arrays
 - Arrays that persist after a function returns
 - Amount of memory is unknown at compile time
 - *Technically possible in stack via VLAs
 - But we're ignoring this;^)

Stack

- Stack
 - Static (compile time) memory allocation
 - Examples: int x; char arr[20];

- Stack
 - Static (compile time) memory allocation
 - Examples: int x; char arr[20];
- Heap
 - Dynamic (runtime) memory allocation

- Stack
 - Static (compile time) memory allocation
 - Examples: int x; char arr[20];
- Heap
 - Dynamic (runtime) memory allocation
 - Examples: malloc(); realloc(); etc.
 - This is our next lecture :^)