Memory Allocation - Static and Dynamic Memory Allocation in C

Dr. B.S.R.V. Prasad
Department of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Vellore - 632014, TN, India

Memory Allocation:

- Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.
- The memory allocation is done either before or at the time of program execution.
- There are two types of memory allocations:
  - Static Memory Allocaiton (Compile-time)
  - Dyanmaic Memory Allocation (Run-time)

Static Memory Allocation

- Static Memory is allocated for declared variables by the compiler.
- The address can be found using the address of operator and can be assigned to a pointer.
- The memory is allocated during compile time.

Dynamic Memory Allocation

- Memory allocation done at the time of execution(run time) is known as dynamic memory allocation.
- Functions `calloc()` and `malloc()` support allocating dynamic memory.
- In the Dynamic allocation of memory space is allocated by using these functions when the value is returned by functions and assigned to pointer variables.

Dynamic Memory Allocaiton

- Dynamic allocation can be handled in two ways
  - Stack allocation:
    - Restricted, but simple and efficient
  - Heap allocation:
    - More general, but less efficient
    - More difficult to implement

Stock Organization

- Memory is freed in opposite order from allocation.
  alloc(A)
  alloc(B)
  alloc(C)
  free(C)
  free(B)
  free(A)

Stock Organization

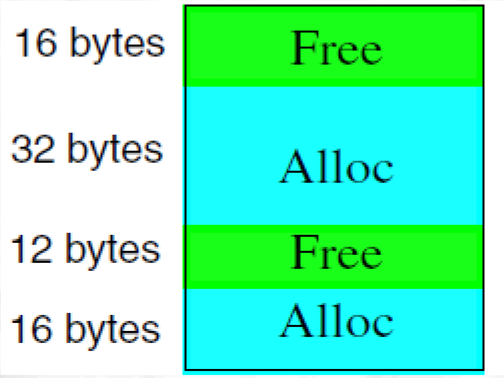
- When is it useful?
  - Memory allocation and freeing are partially predictable
  - Allocation is hierarchical
  - Example
    - Procedure call frames
    - Tree traversal, expression evaluation, parsing

Stack Implementation

- Advance pointer dividing allocated and free space
- Allocate: Increment pointer; Free: Decrement pointer
  alloc(A) alloc(B)
  alloc(C) free(C)
  alloc(D) free(D)
  free(B) free(A)
- **Advantage**
  - Keeps all the free space contiguous
  - Simple and efficient to implement
- **Disadvantage:** Not appropriate for all data structures

Heap Organization

- Allocate from random locations
- Memory consists of allocated areas and free areas (or holes)



16 bytes	Free
32 bytes	Alloc
12 bytes	Free
16 bytes	Alloc

Heap Organization

- When is it useful?
  - Allocation and release are unpredictable
  - Arbitrary list structures, complex data organizations
- Examples: `malloc()` in C
- **Advantage:** Works on arbitrary allocation and free patterns
- **Disadvantage:** End up with small chunks of free space