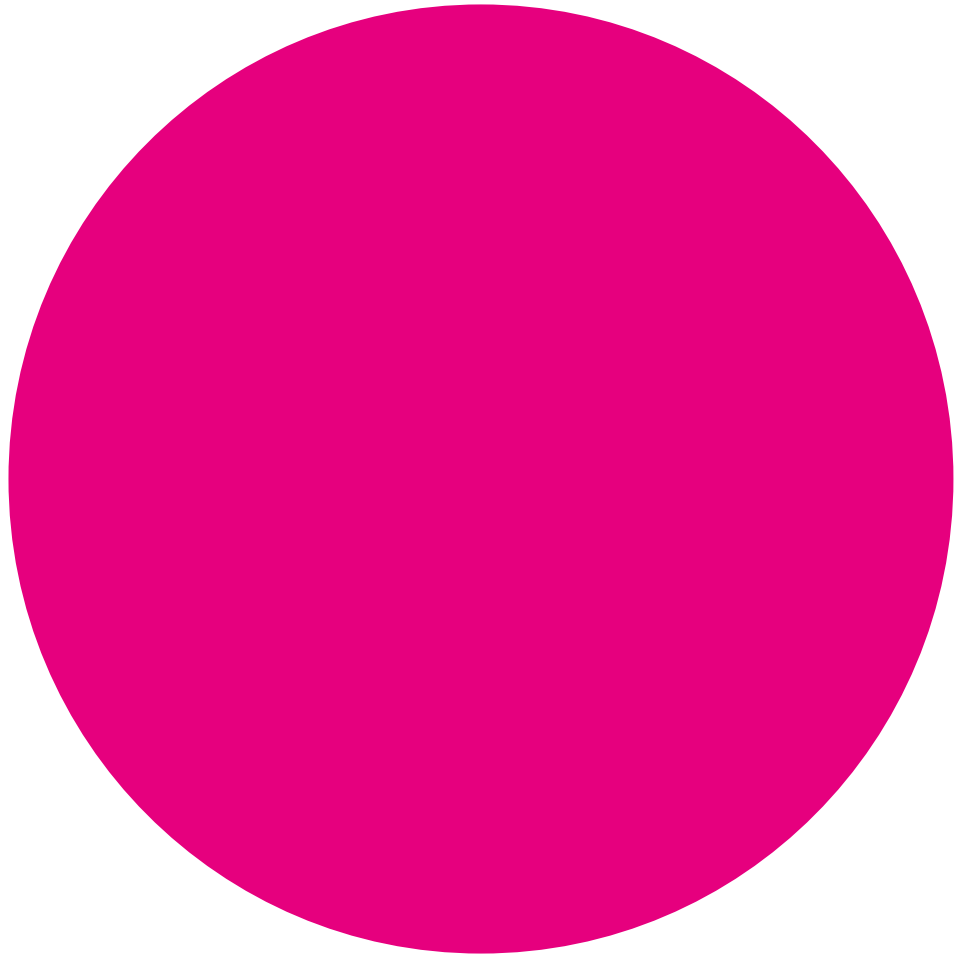
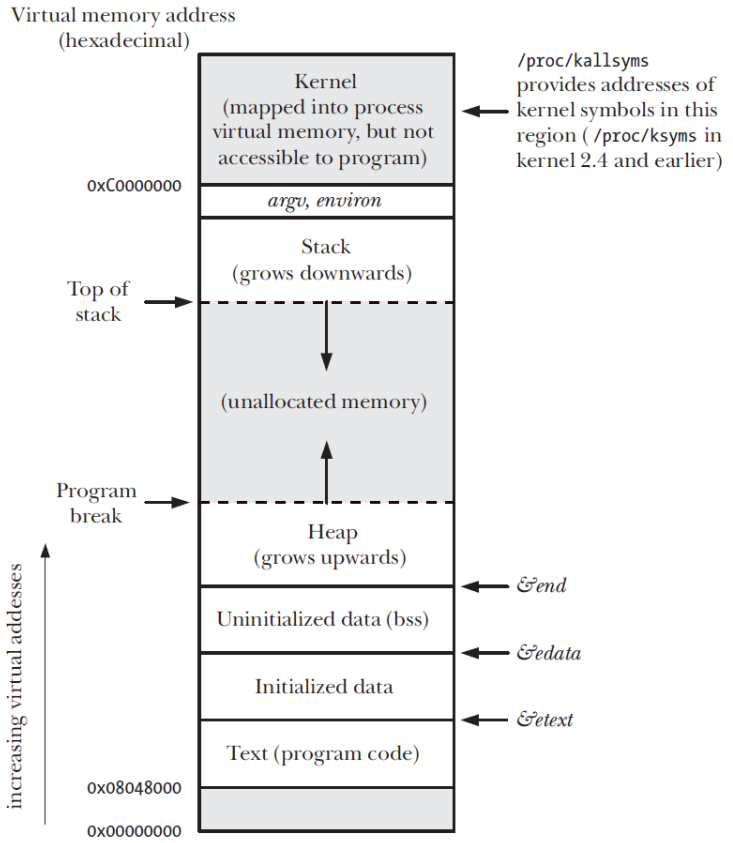
Ch7: Memory Allocation



Heap

- **The heap** is a variable size segment of contiguous virtual memory that begins just after the uninitialized data segment of a process and grows and shrinks as memory is allocated and freed.
- **Program Break** is The current limit of the heap.
- **Adjusting Program break** using brk() and sbrk(). Kernel allocates pages when the new memory is accessed.
- Program break upper and lower limits (end of data, shared memory, process limits).
- Getting the current program break (sbrk(0)).

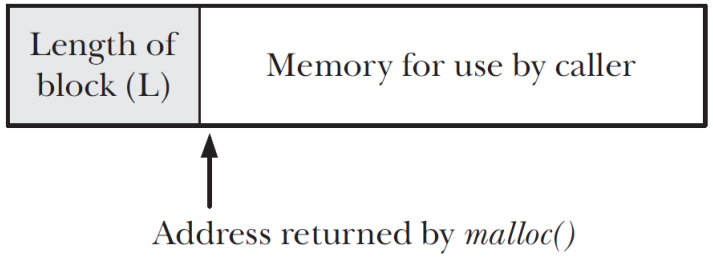


Allocating Memory on the Heap: malloc() and free()

- **malloc()** allocates the needed size and may adjust the program break.
- **free()** usually will not lower the program break.
  - Freed memory might be in the middle of the heap.
  - Minimize the number of sbrk() calls.
  - Programs tend to repeatedly release and reallocate memory.
- When a process terminates, all of its memory is returned to the system. Should we call free()?
  - Readability and maintainability.
  - Long-running programs (Daemons and shells).

Implementation of malloc()

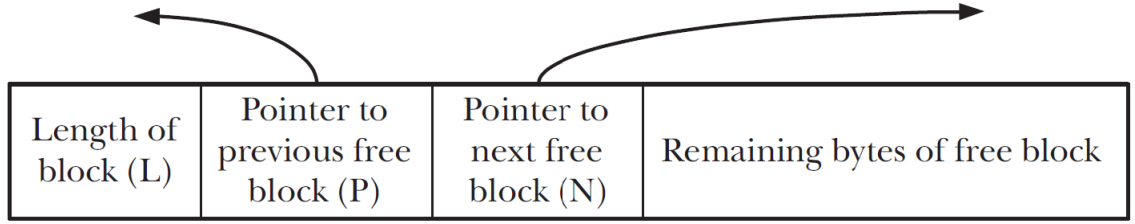
- → Search the free blocks for a good candidate (first-fit, best-fit, etc.).
  - → Split the block if its size is larger than needed.
- → Call sbrk() if no free block matching (with larger size).
- → How free() will know the size of the block?



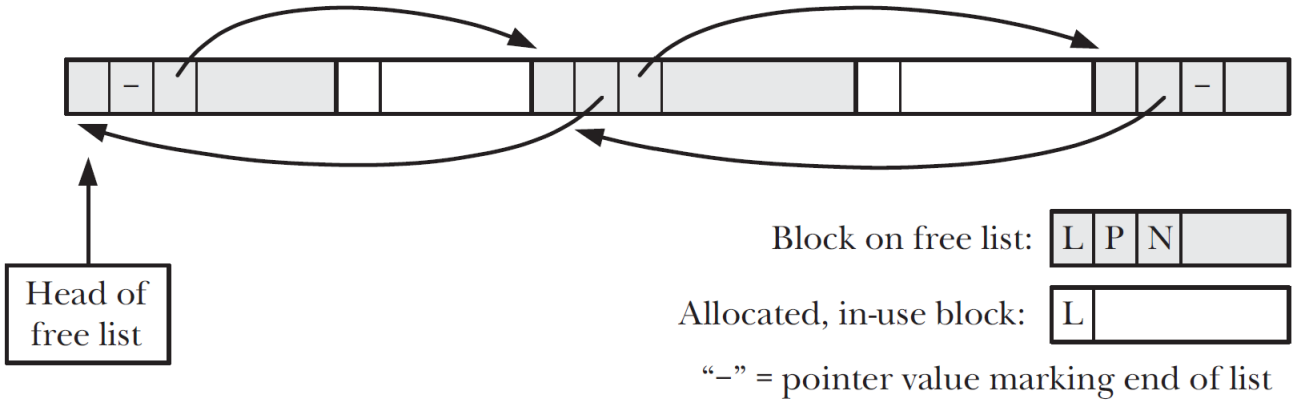
Implementation of free()

→ Get the block length allocated and set by malloc().

→ Construct the free blocks list.



→ The free list will become intermingled with blocks of allocated, in-use memory:



life.augmented

Calloc() and realloc()

- → **calloc()** allocates memory for an array of identical items and initializes the memory to zero.
- → **realloc()** resizes a block of allocated memory.
  - → Attempts to coalesce the block with an immediately following block of memory.
  - → Might copy all existing data from old to new block.
  - → Must use the returned pointer.
  - → Do not assign directly as realloc might fail.

Surviving Rules in Dynamic memory Allocation

- → DO NOT touch any memory outside the allocated block range.
- → DO NOT free an allocated block twice.
- → Free with the same pointer returned from malloc. NOT with offset.
- → Free the allocated memory.

Allocating Memory on the Stack

- **alloca()**
  - Allocates memory dynamically on the stack.
  - Obtains memory from the stack by increasing the size of the stack frame.
  - DO NOT call free().
  - Allocated memory is valid only within the function.