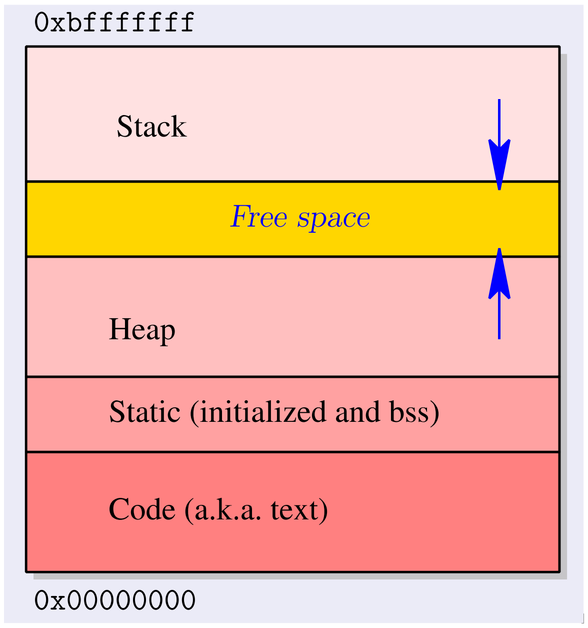
Memory Management

Memory Address Space of a Process



Memory for a process is allocated and initialized when loading and executing a program. Memory access in user mode is restricted to this address space. This address space consists of the following four segments:

1. Code (also called text) segment: .o and executable code
2. Static Data segments: Initialized global (and C static) variables and uninitialized global variables that are zeroed when initializing the process, also called bss
3. Stack segment: Stack frames of function call arguments and local variables, also called automatic variables in C
4. Heap segment: Dynamic allocation (malloc())

System Calls

#include <unistd.h>

*int brk(void *end_data_segment);*
*void *sbrk(intptr_t displacement);*

- brk() sets the end of the data segment, which is also the end of heap to the value specified by *end_data_segment*, when that value is reasonable, the system does have enough memory and the process does not exceed its max data size (see man pages of setrlimit( ) and getrlimit( ))
- sbrk() adds a displacement (possibly 0) and returns the starting address of the new area (it is a C function, front-end to sbrk())

- Both brk( ) and sbrk( ) extend heap. brk(*b*) sets the end of the heap to *b*, while sbrk(n) extends the end of the heap by n bytes. sbrk(0) returns the virtual address just past the end of the heap.
- Both functions are deprecated as "programmer interface" functions, i.e., they are meant for kernel development only.

*void * mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset)*

map length bytes beginning at offset into file fd, preferably at address start (hint only), prot = R/W/X/no access, flags = map_fixed, map_shared, map_private
returns pointer to mmap'ed area

- mmap( ) creates a new mapping in the virtual address space of the calling process
- map length bytes beginning at offset into file fd, preferably at address start (hint only), prot = R/W/X/no access, flags = map_fixed, map_shared, map_private. Returns pointer to mmap'ed area

Read man pages of these functions for the details.

<u>malloc( )</u>

- Calls sbrk( ) to get the memory to allocate in the heap
- malloc is more efficient than allocating memory using brk( ) or sbrk( )
  - malloc( ) does buffering
  - A malloc ( ) call does not always invoke sbrk( )
  - When it calls sbrk() it calls it to allocate a much larger memory than needed

<u>Virtual Memory</u>

- getconf PAGESIZE
- getpagesize( )
- pmap command: memory map of a process
- getrlimit( ) and setrlimit( )
  - Process virtual memory size limit
  - Max CPU time
  - Max data segment size
  - Max file size
  - …