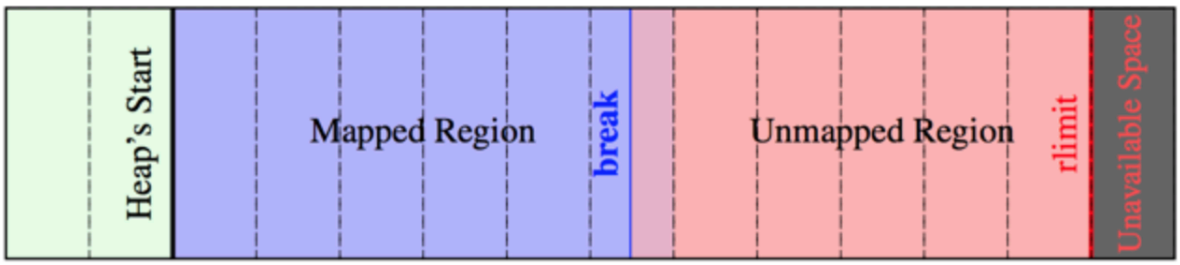
Lab 4: Process (cont)

In which cases we should use aligned_malloc() instead of standard malloc?

We use aligned_function(specially aligned_malloc() and aligned_free) for data alignment. With class object and struct, allocation and manage memory is different from allocation for common variables. By handling the issue about data alignment, we can improve performance of the system a lot. Also that we can use the aligned_malloc() in case the address of a block returned by malloc in GNU systems is always a multiple of eight (or sixteen on 64-bit systems). If we need a block which its address is a multiple of power of two, use aligned_malloc.

How can we increase the size of heap in a running process?

We can easily increase the size of heap in a running process. Heap is a kind of memory. The heap size is comtrolled and managed by heap manager. Some heap manager do not give user anything to control the heap size. But some heap manager give the user some function to allow them to allocate an expand the heap size. We can also use malloc() or calloc() to dynamic allocate more memory. Or we can also write our own allocated function by using sbrk() and brk() system call.



Heap have 3 main boundaries: Heap's start point, break point and rlimit point. These boundaries devide Heap into 2 area: Mapped Region (memory address space already used) and Unmapped region (memory address space haven't been used).We can image that there is a wall between Mapped region and Unmapped region, that is the boundary between them. And when we use system call brk() or sbrk() mean we push that wall forward to the unmapped region to notice to kernel that we are using that memory spaces. In case that heap memory space is run out of space, kernel will use the unavailable space to expand the memory space for user.

Talk about data segments of process, there are (commonly) five segment including kernel space , stack segment, heap segment, text segment, data segment (uninitialized data segment), BSS segment (initialized data segment).Text segment contains code of all program which process is running and this segment is Read-Only. Initialized data segment contains explicitly initialized global and static variables. Uninitialized data segment contains unexplicitly initialized global and static variables. Stack segment contains local variables, arguemants of functions and return value. Heap segment is a segment which is used to dynamically allocate during runtime.