What is brk() and sbrk()? What do they do?

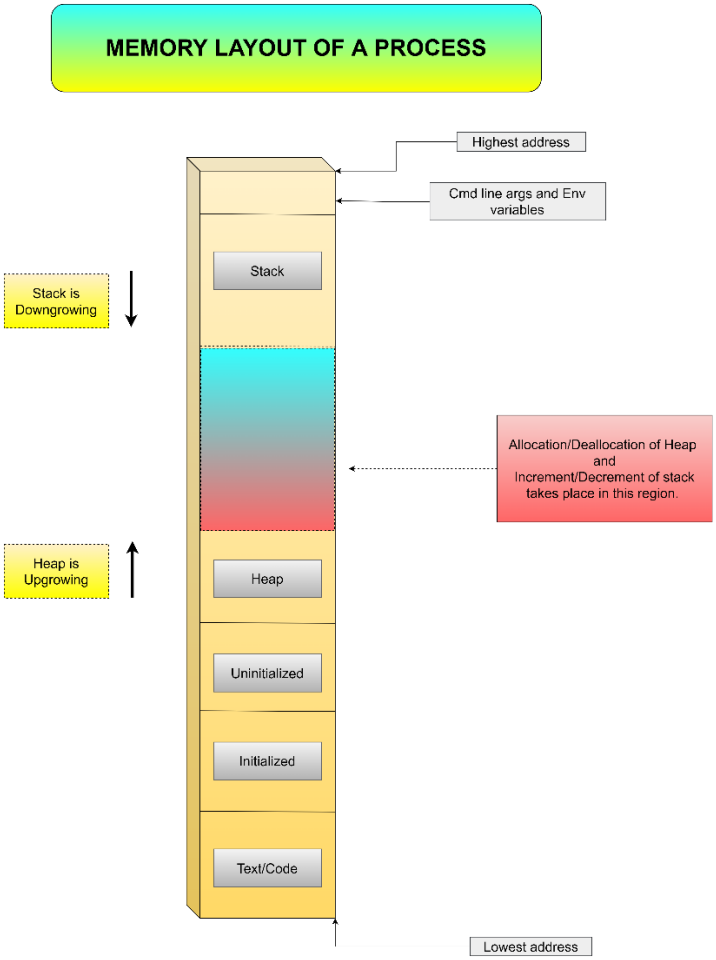
brk() and sbrk() are some primitive functions that are used to change the size of data segment.

As per the man page description:

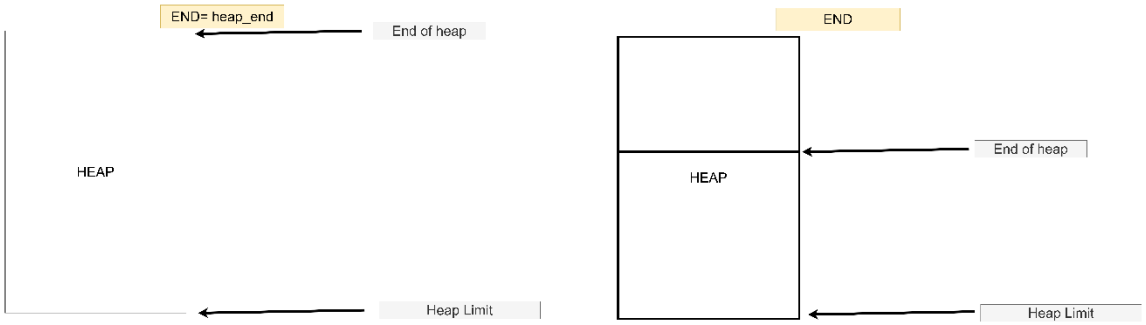
brk() and sbrk() change the location of the program break, which defines the end of the process's data segment (i.e., the program break is the first location after the end of the uninitialized data segment). **Increasing the program break has the effect of allocating memory to the process; decreasing the break deallocates memory.**

brk() sets the end of the data segment to the value specified by addr, when that value is reasonable, the system has enough memory, and the process does not exceed its maximum data size

sbrk() increments the program's data space by increment bytes. Calling sbrk() with an increment of 0 can be used to find the current location of the program break.



You can simply imagine it as a line that moves in the heap memory. The point at which this line is present would be considered at the end of the heap.



The brk() function is used to change the space allocated for the calling process. The amount of allocated space increases as the break value increases and decreases as it does. The newly allocated space is set to 0.

It is true that malloc uses a combination of brk(), sbrk() and mmap(), but the storage space allocated by brk() and sbrk() is different from the storage space allocated by dynamic memory functions such as malloc(), calloc(), realloc().

Because this storage space must be a continuous segment of storage, it is allocated from the initial heap segment only and thus is limited to the initial heap size specified for the calling program or the largest contiguous segment of storage available in the initial heap at the time of the first brk() or sbrk() call. Since this is a separate segment of storage, the brk() and sbrk() functions can be used by an application that is using the other memory allocation functions. However, it is possible that the user's region may not be large enough to support extensive usage of both types of memory allocation.

The brk() function is not portable and cannot be used in multithreaded applications. And hence, usage of mmap is preferred in such cases.

How the brk() and sbrk() functions work:

1) The brk and sbrk calls dynamically change the amount of space allocated for the data segment of the calling process.
2) The change is made by **resetting** the **program break** of the process, which determines the maximum space that can be allocated.
3) The program break is the **address of the first location** beyond the current end of the data region. The amount of available space increases as the break value increases.
4) The **available space is initialized** to a value of **zero**, unless the break is lowered and then increased, as it may reuse the same pages in some unspecified way.
5) The break value can be **automatically rounded up** to a size appropriate for the memory management architecture.

sbrk is used to adjust the program break value by adding a possibly negative size, while brk is used to set the break value to the value of a pointer. Set increment parameter to zero to fetch the current value of the program break.

Upon successful completion, the brk subroutine returns a value of 0, and the sbrk subroutine returns the prior value of the program break (if the available space is increased then this prior value also points to the start of the new area). If either subroutine is unsuccessful, a value of −1 is returned and the errno global variable is set to indicate the error.

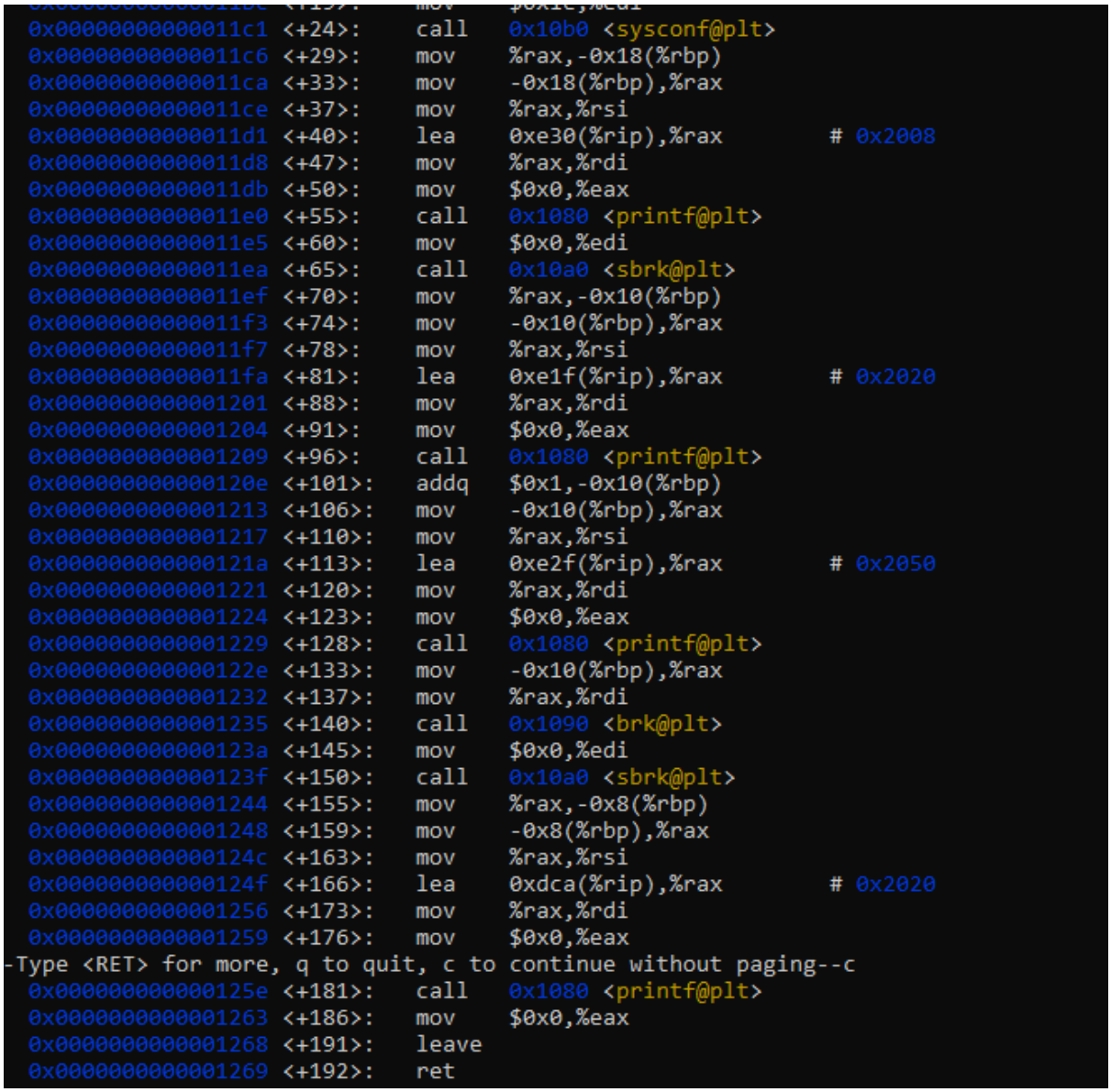
The brk() function asks the kernel to read/write a contiguous chunk of memory from the heap.

If successful, the brk() returns 0. If unsuccessful it returns -1 and sets errno to respective values.

As the break value rises, so does the quantity of allocated space. The newly allocated space has a value of 0 assigned to it. The values of the reallocated space are not emptied if the program first decrements and afterwards increments the break value.

Img1:

```
The page size is: 4096
The current program address break is at: 0x7fffbb4d6000
Point 1 after incrementing: 0x7fffbb4d6001
The current program address break is at: 0x7fffbb4d6001
```



(Address will different as this is a different program)