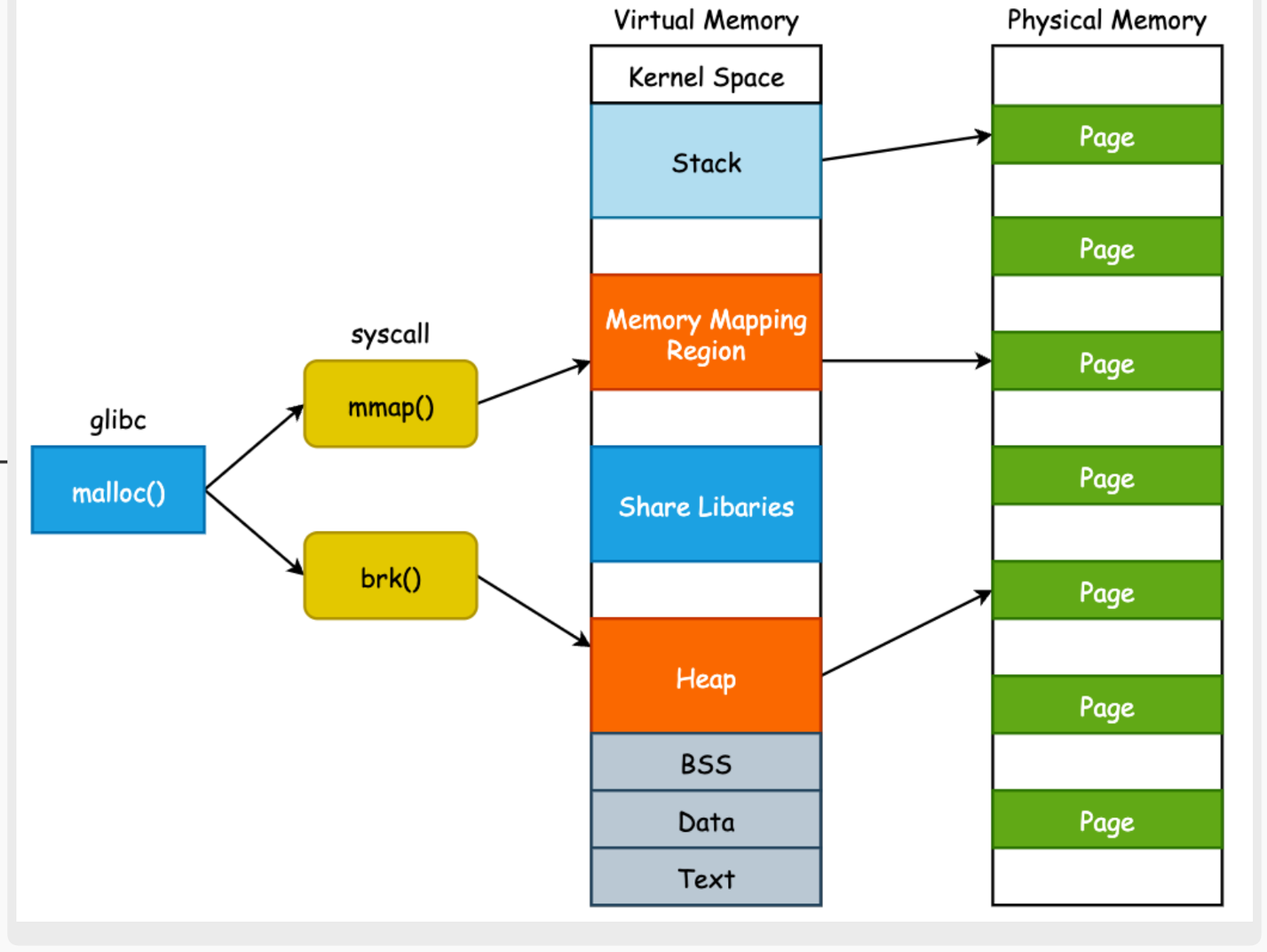
内存申请与内存分配

malloc() 概览

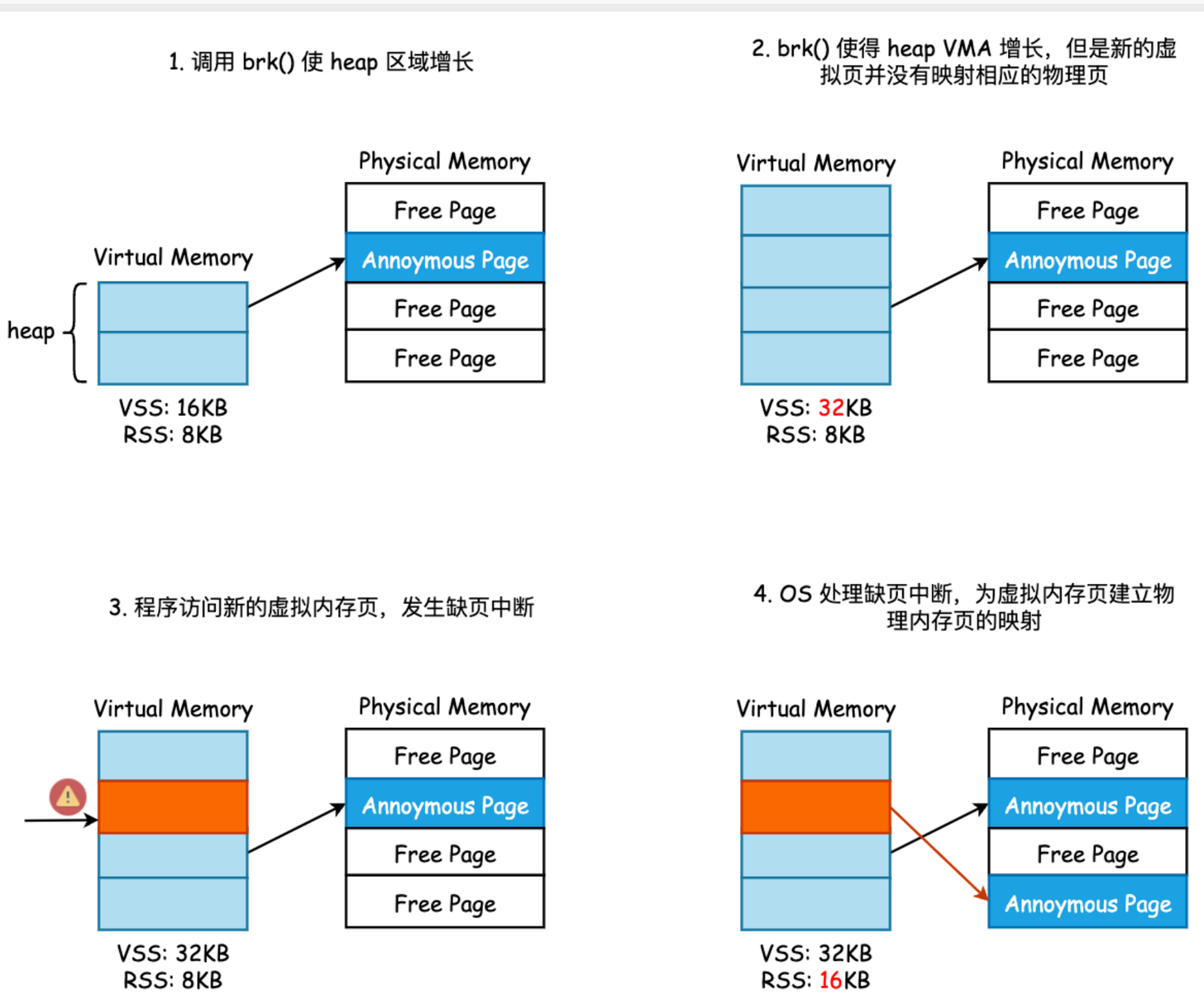
首先需要明确的是，malloc() 是 glibc 一个库函数，并不是系统调用。malloc() 本质上其实就是封装了 brk() 以及 mmap() 等系统调用，同时在内部进行了一些内存管理



如上图所示，malloc() 的本质就是在其实现内部通过调用 brk() 或者是 mmap() 来改变进程的虚拟内存空间，从而给进程分配虚拟内存的。而虚拟内存到物理内存之间的映射，由操作系统管理，用户进程无法干预此过程

VSS vs RSS

VSS，即 Virtual Set Size，表示进程所使用的虚拟内存大小；RSS，即 Resident Set Size，即进程所使用的物理内存大小。进程的 VSS 和 RSS 可以通过 top/htop 命令进行查看，对应结果中的 VIRT 以及 RES



因此，我们一定要明确，使用 malloc() 申请虚拟内存，并不代表着物理内存的分配，只有应用程序访问了申请的虚拟内存区域时，操作系统才会为进程分配物理内存页

Experiment

我们可以通过将 overcommit_memory 设置为 1 来允许进程申请不超过虚拟内存大小的内存，从而观察内存申请与内存分配之间的区别

```
sudo swapoff -a
sudo sh -c 'echo 1 >/proc/sys/vm/overcommit_memory'
```

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    for (int i = 0; i < 200000; i++) {
        char *p = malloc(1024 * 1024 * 1024);
        if (p == NULL) {
            printf("malloc failed! \n");
            exit(1);
        }
        else {
            printf("malloc success: %d \n", i);
        }
    }
}
```

运行上面的简易程序就会发现，我们使用 malloc() 能够成功申请的内存数量要远超于物理内存的大小

虚拟内存和物理内存之间的映射，本质上就是一种 Lazy Load，或者说懒加载，对提高系统性能和负载有着至关重要的作用