VSS- Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）

RSS- Resident Set Size 实际使用物理内存（包含共享库占用的内存）

PSS- Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存，把共享库按比例分配后的实际使用内存）

USS- Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）

一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS

在Linux里面，一个进程占用的内存有不同种说法，可以是VSS/RSS/PSS/USS四种形式，这四种形式首字母分别是Virtual/Resident/Proportional/Unique的意思。

VSS是单个进程全部可访问的地址空间，其大小可能包括还尚未在内存中驻留的部分。对于确定单个进程实际内存使用大小，VSS用处不大。

RSS是单个进程实际占用的内存大小，RSS不太准确的地方在于它包括该进程所使用共享库全部内存大小。对于一个共享库，可能被多个进程使用，实际该共享库只会被装入内存一次。

进而引出了PSS，PSS相对于RSS计算共享库内存大小是按比例的。N个进程共享，该库对PSS大小的贡献只有1/N。

USS是单个进程私有的内存大小，即该进程独占的内存部分。USS揭示了运行一个特定进程在的真实内存增量大小。如果进程终止，USS就是实际被返还给系统的内存大小。

综上所属，VSS>RSS>PSS>USS(等于毫就不写了)。

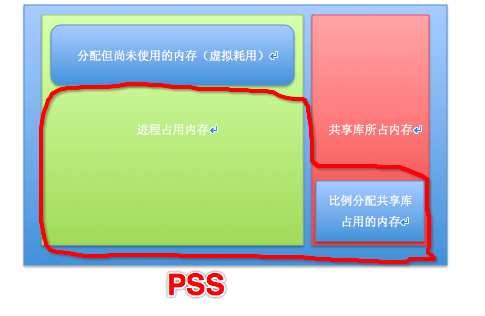
可以看出VSS只是一个虚拟空间大小，对内存实际占用量意义不大。

RSS是对于计算一个进程内存占用量，会有一点误解。因为像libc这种大部头库文件，共享者很多，都算在一个进程头上不科学。

这时候PSS就更加科学了，除了自己独占的内存，再加上分到的共享部分。

USS在计算一个新加入的进程导致系统内存增量很有用处，因为共享部分已经存在，并不是由其导致的。

一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS

VSS - Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）  
RSS - Resident Set Size 实际使用物理内存（包含共享库占用的内存）  
PSS - Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）  
USS - Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）  
[](http://img4.tbcdn.cn/L1/461/1/d3a92df3efa0df779418bed820e6dcd31f6cbbc6)  
[](http://img4.tbcdn.cn/L1/461/1/d10c3c6e80e70309ce73bfa874d92d56606fa989)  
[](http://img1.tbcdn.cn/L1/461/1/ee8a35925f0aafb160c58b18eb4aee3bf4762398)  
[](http://img2.tbcdn.cn/L1/461/1/255c513123c88d85d5e1137be66d0672368b8931)

很多人通过累加 “ps  aux” 命令显示的 RSS 列来统计全部进程总共占用的物理内存大小，这是不对的。RSS(resident set size)表示常驻内存的大小，但是由于不同的进程之间会共享内存，所以把所有进程RSS进行累加的方法会重复计算共享内存，得到的结果是偏大的。  
  
  
正确的方法是累加 /proc/[1-9]\*/smaps 中的 Pss 。/proc/<pid>/smaps 包含了进程的每一个内存映射的统计值，详见proc(5)的手册页。Pss(Proportional Set Size)把共享内存的Rss进行了平均分摊，比如某一块100MB的内存被10个进程共享，那么每个进程就摊到10MB。这样，累加Pss就不会导致共享内存被重复计算了。  
  
  
命令如下：  
$ grep Pss /proc/[1-9]\*/smaps | awk '{total+=$2}; END {print total}'

上面的[1-9]\*就是所有的进程的id号。如果想要查看某一个进程的占用的内存的多少，就可以把[1-9]\*替换成进程的id号。  
需要注意的是，全部进程占用的内存并不等于 free 命令所显示的 “used memory”，因为“used memory”不仅包含了进程所占用的内存，还包含cache/buffer以及kernel动态分配的内存等等。  
  
# grep Pss /proc/[1-9]\*/smaps | awk '{total+=$2}; END {print total}'  
2296454  
   
# free -k  
              total        used        free      shared  buff/cache   available  
Mem:        3809036     2441052      248504      183808     1119480      861748  
Swap:       4063228      866856     3196372  
有人提出【MemTotal = MemFree + buff/cache + slab + 全部进程占用的内存】。这是不对的，原因之一是：进程占用的内存包含了一部分page cache，换句话说，就是进程占用的内存与page cache发生了重叠。比如进程的mmap文件映射同时也统计在page cache中。