Linux任督二脉之内存管理(三)

麦当劳喜欢您来，喜欢您再来

进程的内存消耗和泄漏

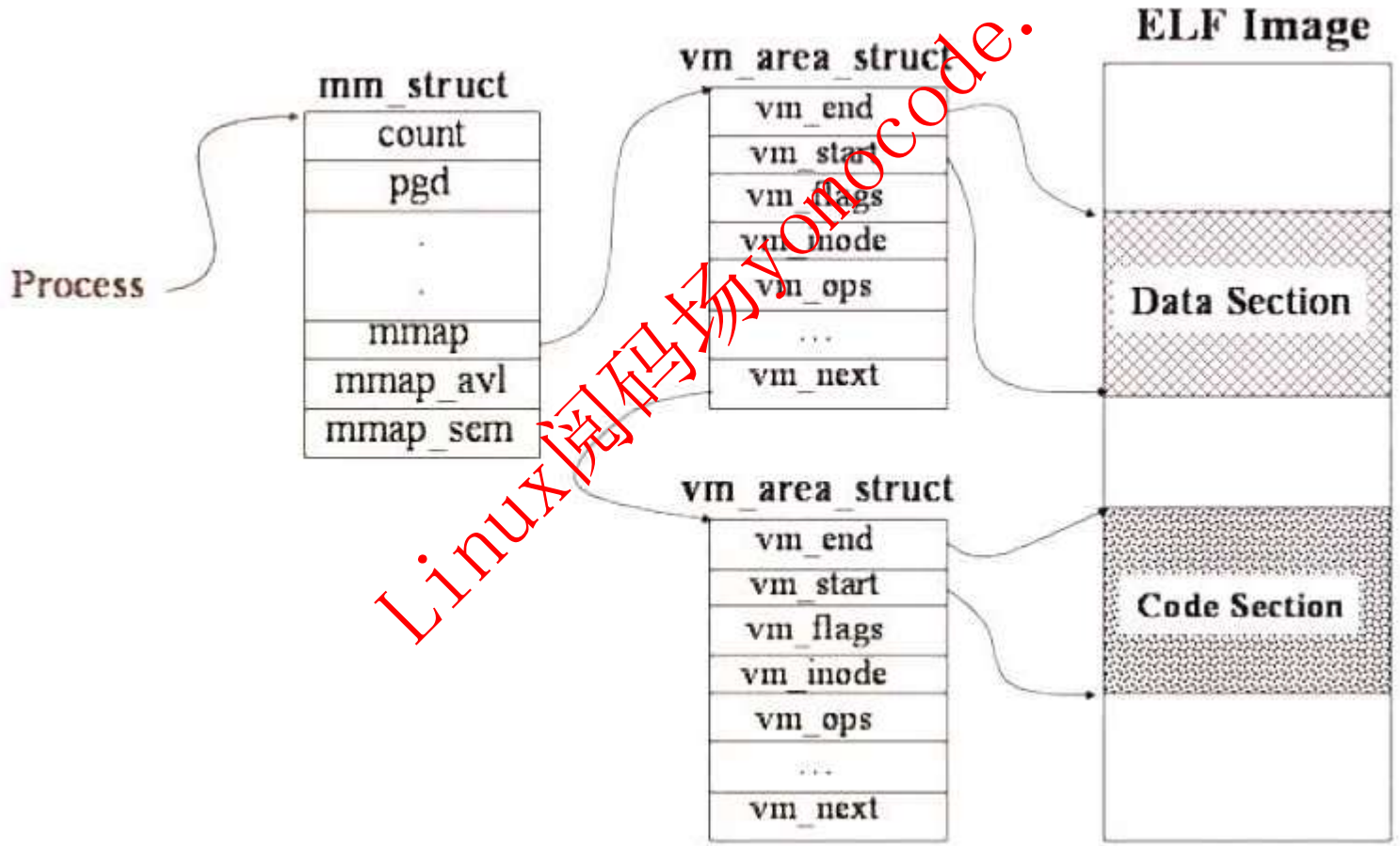
*进程的VMA。
*进程内存消耗的4个概念：vss、rss、pss和uss
*page fault的几种可能性，major和minor。
*应用内存泄漏的界定方法
*应用内存泄漏的检测方法：valgrind和addresssanitizer

练习题

*看一下进程的/proc/<pid>/maps和smaps文件；
*pmap一个进程；
*把同一个程序运行2次，运行1次，观察pss；再运行，得到2个进程，观察代码段的pss变化；
*valgrind检查内存错误

进程的虚拟地址空间VMA

进程的每一段虚拟地址空间就是一个VMA



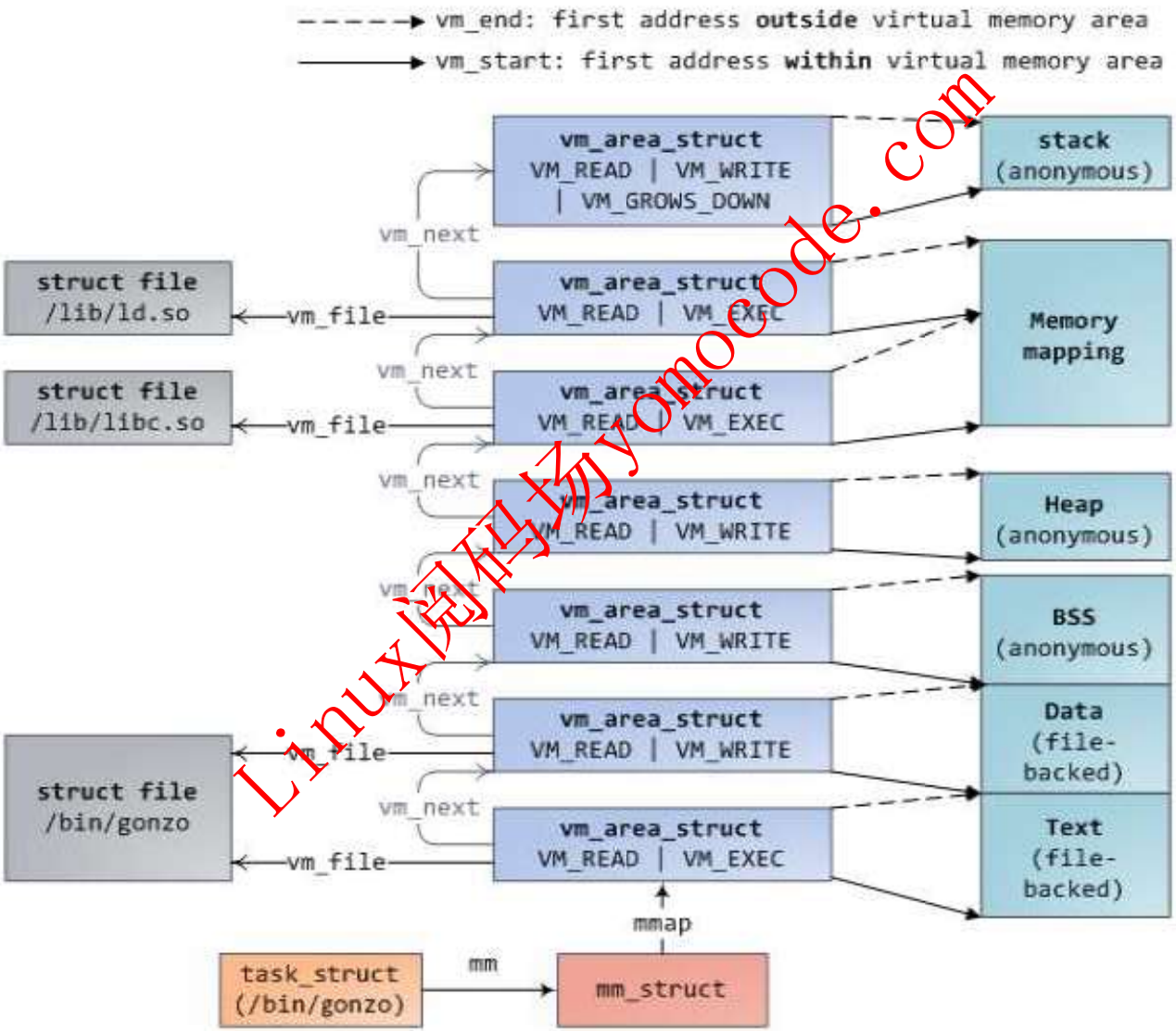
pmap, /proc/<pid>/maps, /proc/<pid>/smaps

```
baohua@baohua-VirtualBox:~$ pmap 3474
3474:   ./a.out
08048000      4K r-x-- a.out
08049000      4K r---- a.out
0804a000      4K rw--- a.out
b75e0000      4K rw---   [ anon ]
b75e1000   1704K r-x-- libc-2.19.so
b778b000      8K r---- libc-2.19.so
b778d000      4K rw--- libc-2.19.so
b778e000     12K rw---   [ anon ]
b77af000     12K rw---   [ anon ]
b77b2000      8K r----   [ anon ]
b77b4000      8K r-x--   [ anon ]
b77b6000    128K r-x-- ld-2.19.so
b77d6000      4K r---- ld-2.19.so
b77d7000      4K rw--- ld-2.19.so
bfcbf000    132K rw---   [ stack ]
 total     2040K
```

```
baohua@baohua-VirtualBox:~$ cat /proc/3474/maps
08048000-08049000 r-xp 00000000 08:01 265913     /home/baohua/a.out
08049000-0804a000 r--p 00000000 08:01 265913     /home/baohua/a.out
0804a000-0804b000 rw-p 00001000 08:01 265913     /home/baohua/a.out
b75e0000-b75e1000 rw-p 00000000 00:00 0
b75e1000-b778b000 r-xp 00000000 08:01 560893     /lib/i386-linux-gnu/libc-2.19.so
b778b000-b778d000 r--p 001aa000 08:01 560893     /lib/i386-linux-gnu/libc-2.19.so
b778d000-b778e000 rw-p 001ac000 08:01 560893     /lib/i386-linux-gnu/libc-2.19.so
b778e000-b7791000 rw-p 00000000 00:00 0
b77af000-b77b2000 rw-p 00000000 00:00 0
b77b2000-b77b4000 r--p 00000000 00:00 0          [vvar]
b77b4000-b77b6000 r-xp 00000000 00:00 0          [vdso]
b77b6000-b77d6000 r-xp 00000000 08:01 575684     /lib/i386-linux-gnu/ld-2.19.so
b77d6000-b77d7000 r--p 0001f000 08:01 575684     /lib/i386-linux-gnu/ld-2.19.so
b77d7000-b77d8000 rw-p 00020000 08:01 575684     /lib/i386-linux-gnu/ld-2.19.so
bfcbf000-bfce0000 rw-p 00000000 00:00 0          [stack]
```

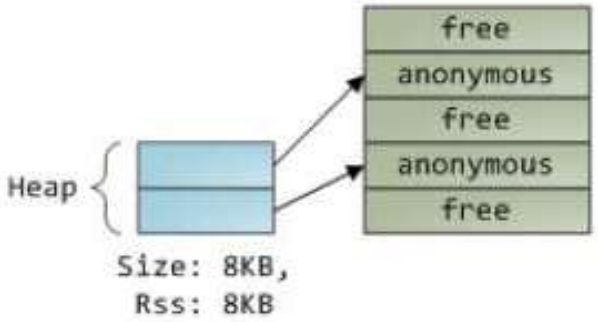
```
baohua@baohua-VirtualBox:~$ cat /proc/3474/smaps  | more
08048000-08049000 r-xp 00000000 08:01 265913     /home/baohua/a.out
Size:                  4 kB
Rss:                   4 kB
Pss:                   4 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         4 kB
Private_Dirty:         0 kB
Referenced:            4 kB
Anonymous:             0 kB
AnonHugePages:         0 kB
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd ex mr mw me dw
08049000-0804a000 r--p 00000000 08:01 265913     /home/baohua/a.out
Size:                  4 kB
Rss:                   4 kB
```

VMA与程序的各个段以及库

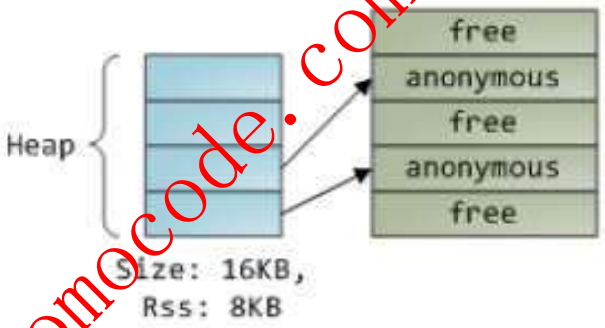


VSS vs. RSS

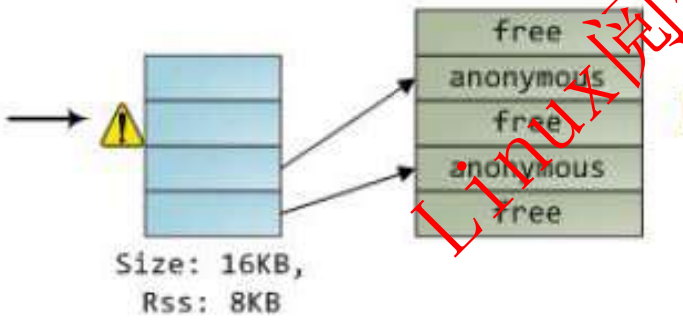
1. Program calls brk() to grow its heap



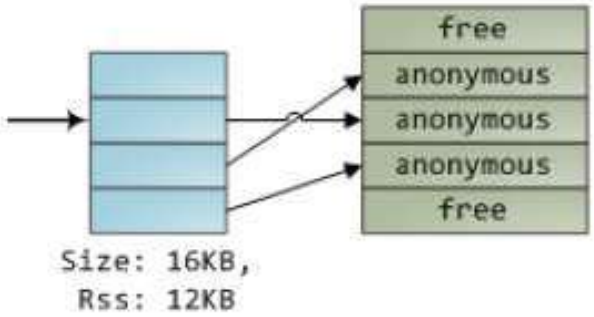
2. brk() enlarges heap VMA.
New pages are **not** mapped onto physical memory.



3. Program tries to access new memory.
Processor page faults.



4. Kernel assigns page frame to process, creates PTE, resumes execution. Program is unaware anything happened.



Page fault的几种可能性

第一次写，发生page fault

申请一页内存

1

内存访问落在非法区，segv 2

在此区写，segv 3

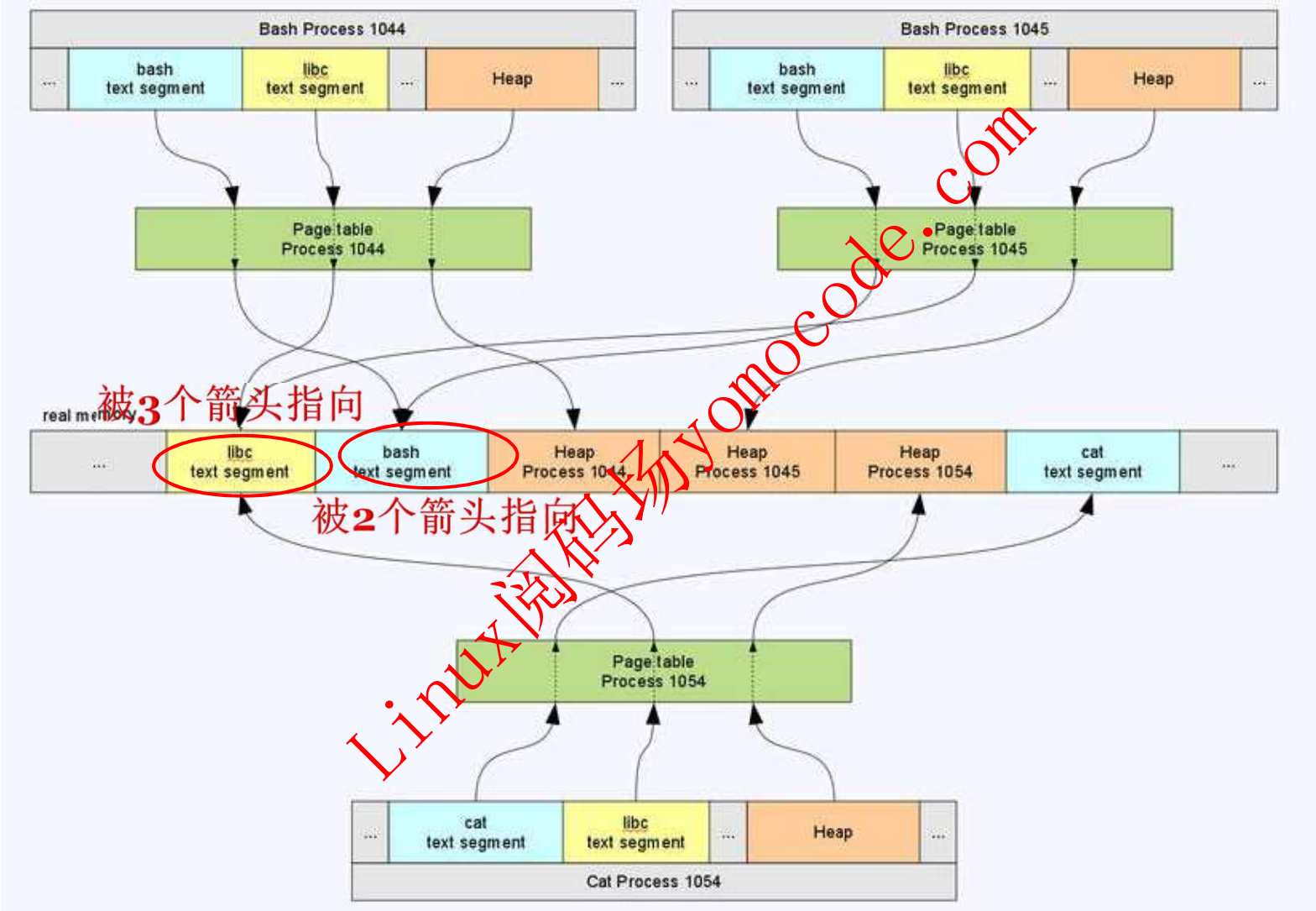
在此区执行，申请页，读出代码段 4

结论：发生pagefault后，有可能segv

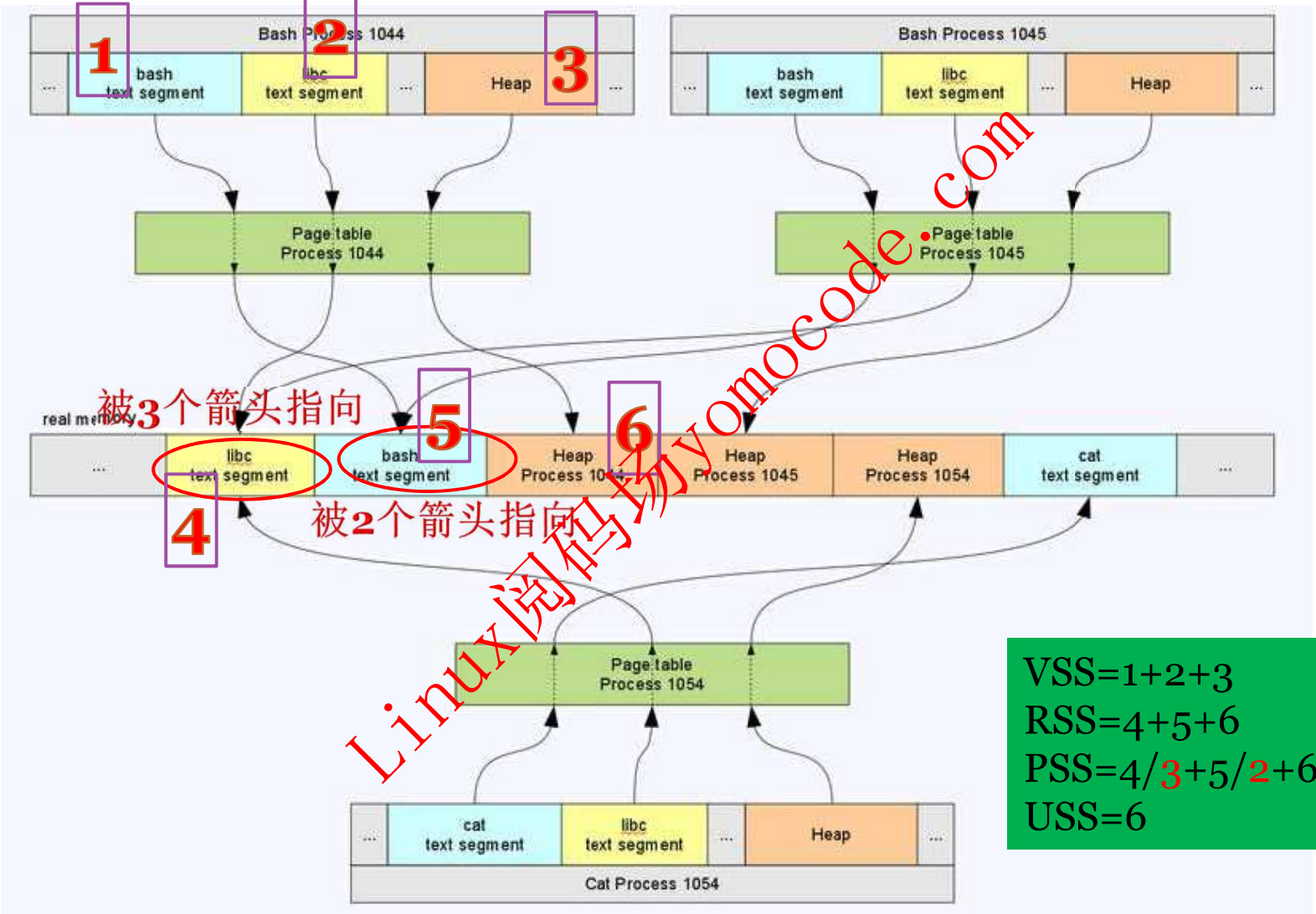
- ✓ 非法区域 (2)
- ✓ VMA权限不对 (3)

有可能，不segv，而是申请内存(1Minor,4Major)

内存被进程如何瓜分?



vss/rss/pss/uss

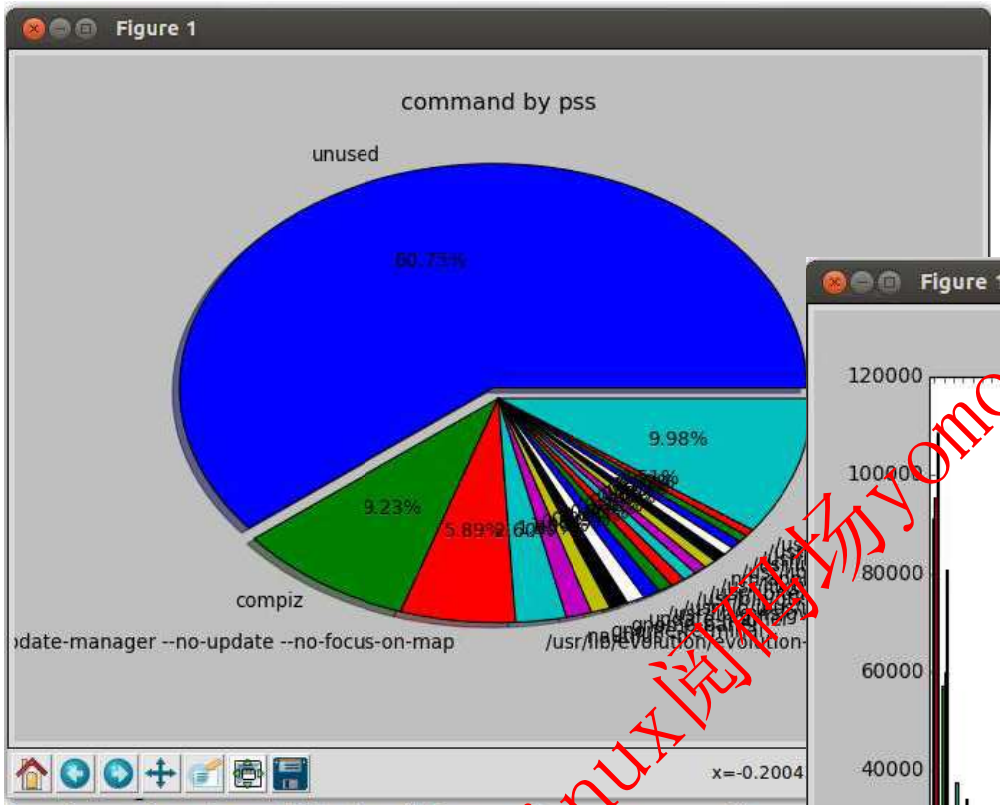


VSS=1+2+3
RSS=4+5+6
PSS=4/3+5/2+6
USS=6

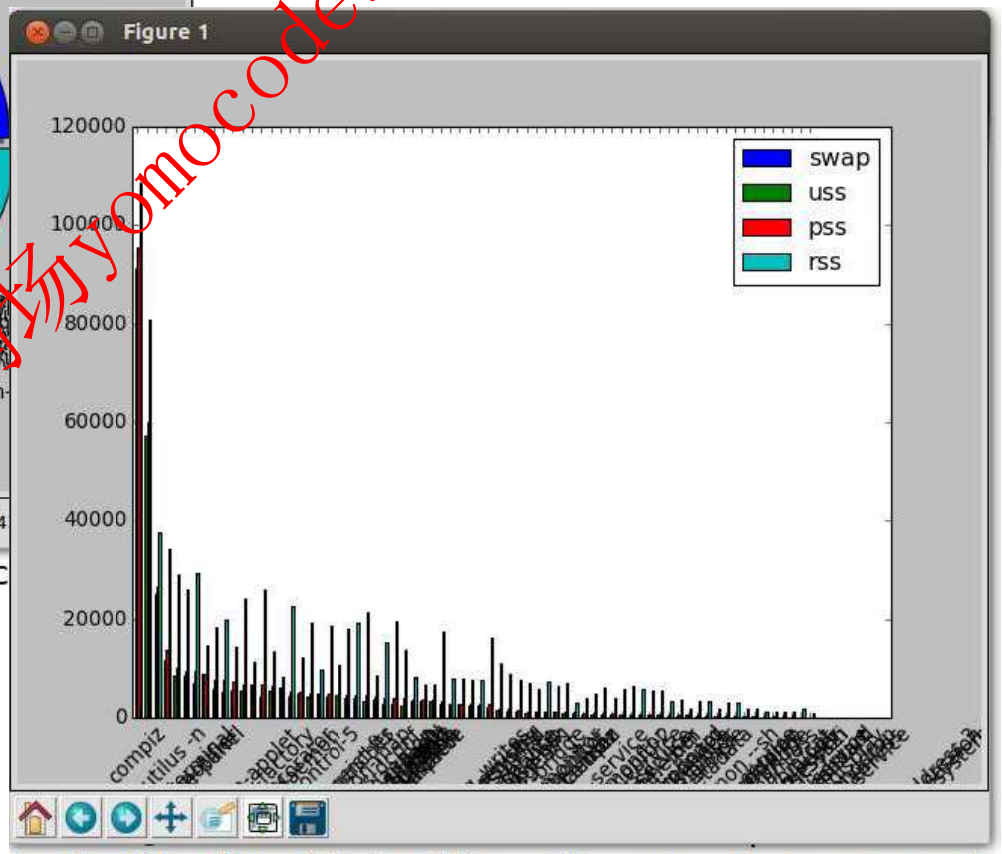
smem – process memory

```
baohua@baohua-VirtualBox:~$ smem
  PID User     Command                         Swap      USS      PSS      RSS
 3474 baohua   ./a.out                            0       68       76     1028
 3352 baohua   /bin/cat                           0      112      143     1896
 2750 baohua   /usr/bin/VBoxClient --seaml        0       84      217     1164
 2743 baohua   /usr/bin/VBoxClient --displ        0       84      219     1172
 2755 baohua   /usr/bin/VBoxClient --draga        0       92      219     1140
 2733 baohua   /usr/bin/VBoxClient --cliph        0      120      236     1160
 2838 baohua   upstart-dbus-bridge --daemo        0      208      271     1696
 2839 baohua   upstart-dbus-bridge --daemo        0      228      295     1784
 2787 baohua   upstart-event-bridge               0      264      315     2928
 3049 baohua   /sbin/initctl emit --indicato      0      232      327     2988
 2836 baohua   upstart-file-bridge --daemo        0      268      346     1772
 2752 baohua   /usr/bin/VBoxClient --seaml        0      244      467     3296
 2746 baohua   /usr/bin/VBoxClient --displ        0      248      474     3384
 2830 baohua   gpg-agent --daemon --sh           32      528      543     1704
 2758 baohua   /usr/bin/VBoxClient --draga       32      332      563     3572
 2887 baohua   /bin/dbus-daemon --config-f        0      404      581     3320
 3054 baohua   /usr/lib/i386-linux-gnu/ind        0      524      602     5484
 3287 baohua   /usr/lib/gvfs/gvfsd-metadat        0      544      618     5580
 2892 baohua   /usr/lib/at-spi2-core/at-sp        0      540      640     5844
 2960 baohua   /usr/lib/ibus/ibus-engine-s        0      556      691     6364
```

smem – pie和bar




```
baohua@baohua-VirtualBox:~$ smem --pie=c
```




```
baohua@baohua-VirtualBox:~$ smem --bar=command
```

进程内存泄漏的界定

连续多点采样法，随着时间越久，进程耗费内存越多

观察一个有内存泄漏的进程

```
void main(void)
{
        unsigned int *p1, *p2;
        while(1)
        {
                p1=malloc(4096*3);
                p1[0] = 0;
                p1[1024] = 1;
                p1[1024*2] = 2;

                p2=malloc(1024);
                p2[0] = 1;
                free(p2);
                sleep(1);
        }
}
```

内存随着时间变大

```
baohua@baohua-VirtualBox:~$ smem -P a.out
  PID User     Command                         Swap      USS      PSS      RSS
 3836 baohua   ./a.out                            0      116      125     1172
 3837 baohua   /usr/bin/python /usr/bin/sm        0     4776     4803     6912
baohua@baohua-VirtualBox:~$ smem -P a.out
  PID User     Command                         Swap      USS      PSS      RSS
 3836 baohua   ./a.out                            0      128      137     1184
 3838 baohua   /usr/bin/python /usr/bin/sm        0     4776     4801     6840
baohua@baohua-VirtualBox:~$ smem -P a.out
  PID User     Command                         Swap      USS      PSS      RSS
 3836 baohua   ./a.out                            0      140      149     1196
 3839 baohua   /usr/bin/python /usr/bin/sm        0     4780     4811     6904
baohua@baohua-VirtualBox:~$ smem -P a.out
  PID User     Command                         Swap      USS      PSS      RSS
 3836 baohua   ./a.out                            0      152      161     1208
 3840 baohua   /usr/bin/python /usr/bin/sm        0     4776     4803     6912
```

内存泄漏的检查-valgrind

valgrind --tool=memcheck --leak-check=yes ./a.out

```
==3978== Memcheck, a memory error detector
==3978== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3978== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==3978== Command: ./a.out
==3978==
^C==3978==
==3978== HEAP SUMMARY:
==3978==     in use at exit: 73,728 bytes in 6 blocks
==3978==   total heap usage: 12 allocs, 6 frees, 79,872 bytes allocated
==3978==
==3978== 61,440 bytes in 5 blocks are definitely lost in loss record 2 of 2
==3978==    at 0x402A17C: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3978==    by 0x8048491: main (leak-example.c:6)
==3978==
==3978== LEAK SUMMARY:
==3978==    definitely lost: 61,440 bytes in 5 blocks
==3978==    indirectly lost: 0 bytes in 0 blocks
==3978==      possibly lost: 0 bytes in 0 blocks
==3978==    still reachable: 12,288 bytes in 1 blocks
==3978==         suppressed: 0 bytes in 0 blocks
==3978== Reachable blocks (those to which a pointer was found) are not shown.
```

内存泄漏的检查-addresssanitizer

gcc -g -fsanitize=address ./leak-example.c

```
./a.out

=================================================================
==5232==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 12288 byte(s) in 1 object(s) allocated from:
    #0 0x7fab49936602 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
    #1 0x4007d7 in main leak-example.c:6
    #2 0x7fab494f482f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)

SUMMARY: AddressSanitizer: 12288 byte(s) leaked in 1 allocation(s).
```

13 17855, 1/19/2018

课程练习源码

https://github.com/21cnbao/memory-courses

谢 谢！