UC Inside, World in Hand!

Linux常用监控命令介绍

基础应用组 梁若羽

2011-07-12

内容

1. 培训目标
2. 基础知识
3. 常用监控命令
4. 在实战中综合运用

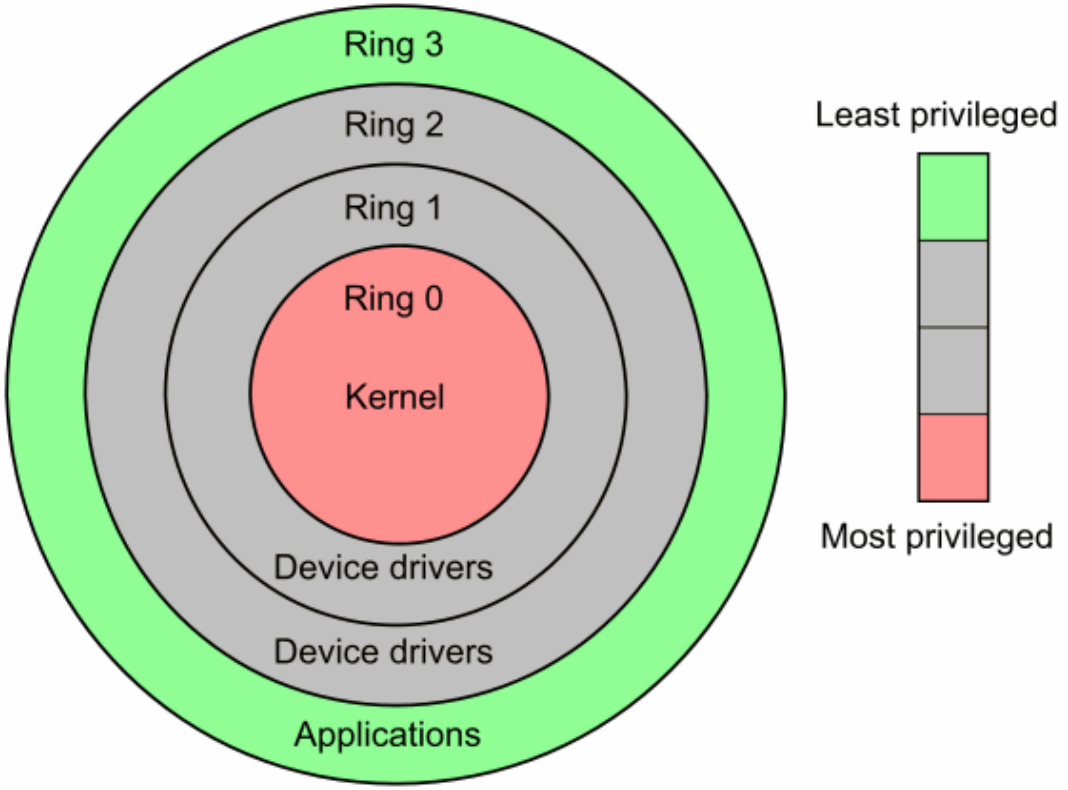
培训目标

- 掌握常用监控命令的用途和启用方法
- 熟悉各个关键输出参数的真实含义
- 了解Linux操作系统的一些基本原理
- 抛砖引玉，交流经验

内容

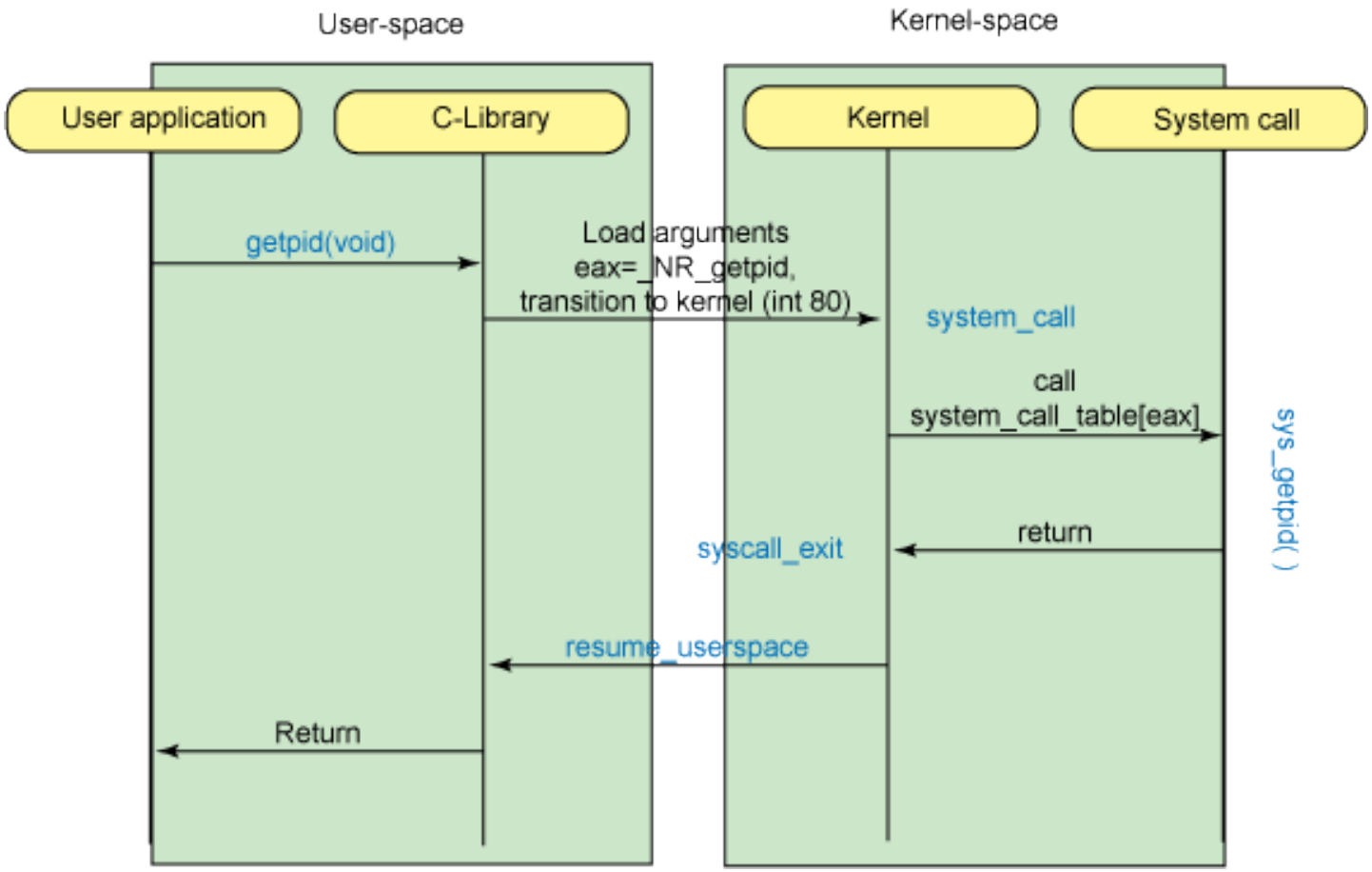
1. 培训目标
2. 基础知识
3. 常用监控命令
4. 在实战中综合运用

基础知识——核心态与用户态



- 核心态：ring 0，操作系统内核代码运行的模式，可以无限制访问系统存储、外部设备
- 用户态：ring 3，应用程序自身代码运行的模式，非特权状态，能够访问的地址空间是有限的

基础知识——系统调用(1)



- 系统调用：核心态开放给用户态的接口，以软中断实现
- 函数调用：用户态层面的概念，与核心态无关

基础知识——系统调用(2)

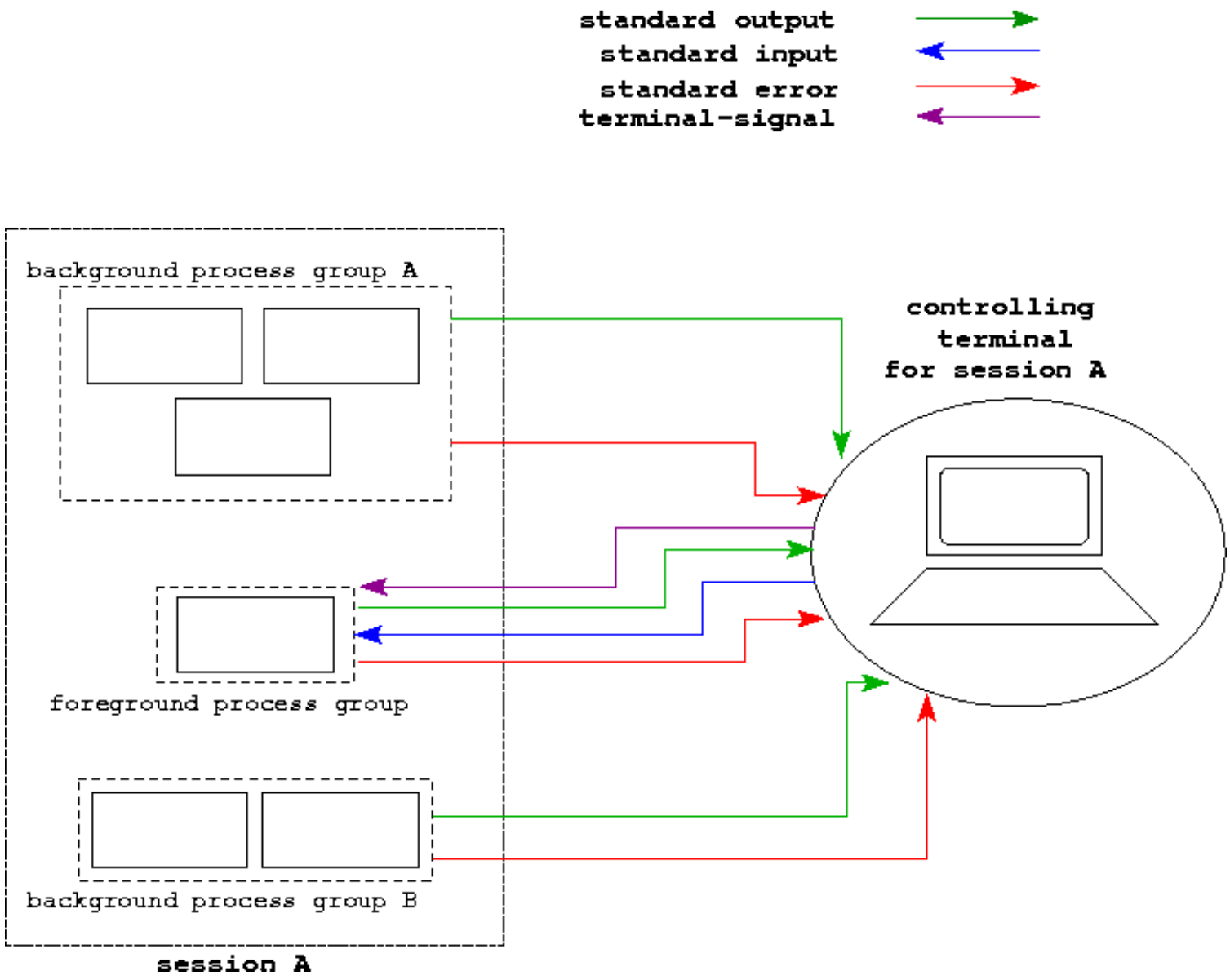
```
int main(void) {
    printf("hello world!");
    return 0;
}
```

$ strace ./helloworld

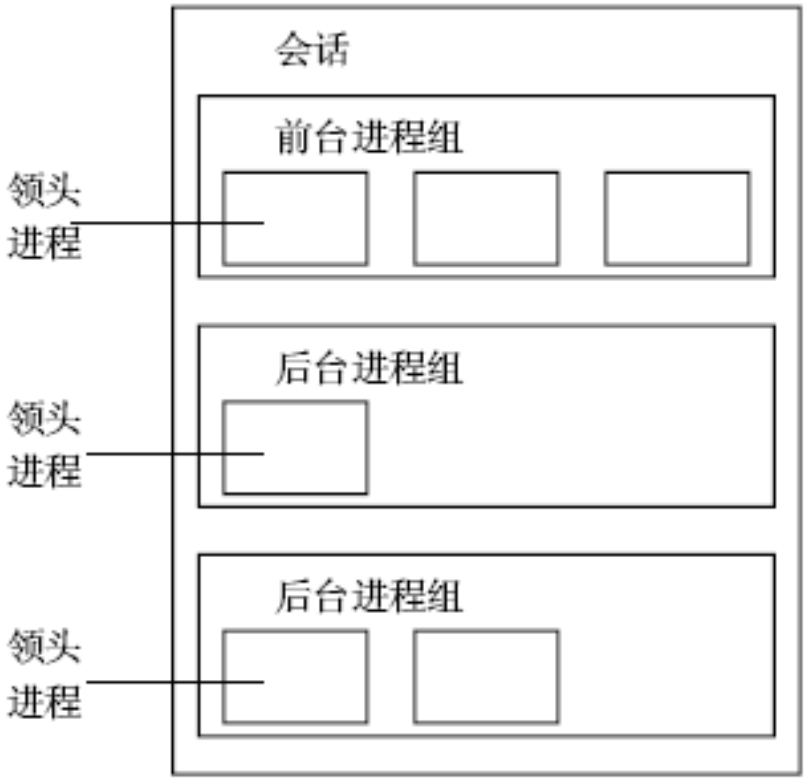
```
...
write(1, "hello world!", 12hello world!) = 12
exit_group(0)                          = ?
```

- 系统调用无处不在，哪怕是最简单的hello world！
- strace命令能够显示所有在程序中使用的系统调用

基础知识——进程组与会话(1)



基础知识——进程组与会话(2)



- 进程组是一个或多个进程的集合，进程组ID是该组的领头进程的pid
- 会话是一个或多个进程组的集合，当前与终端交互的进程组为前台进程组，只能有一个，其余为后台进程组

基础知识——进程的几种状态

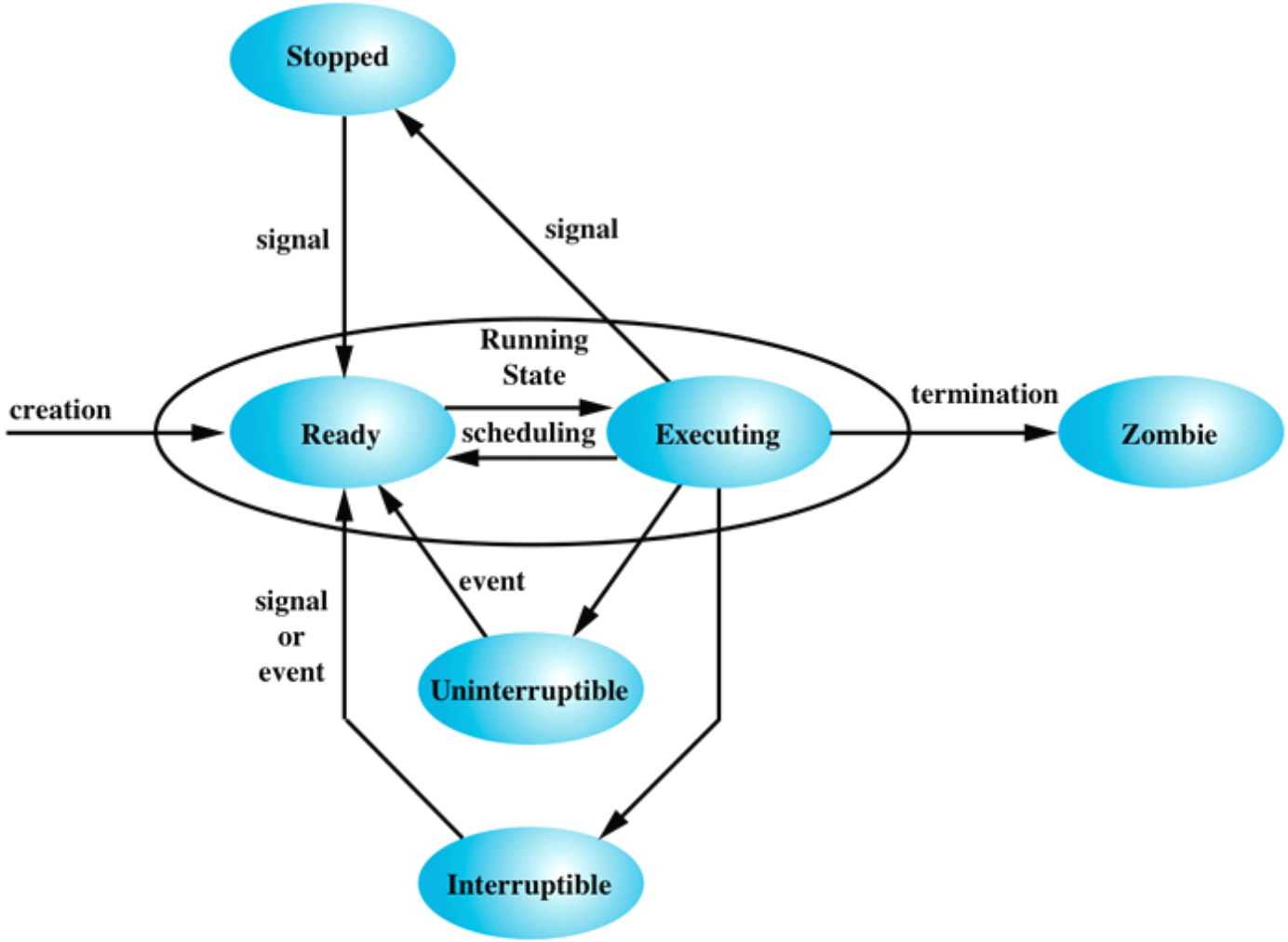
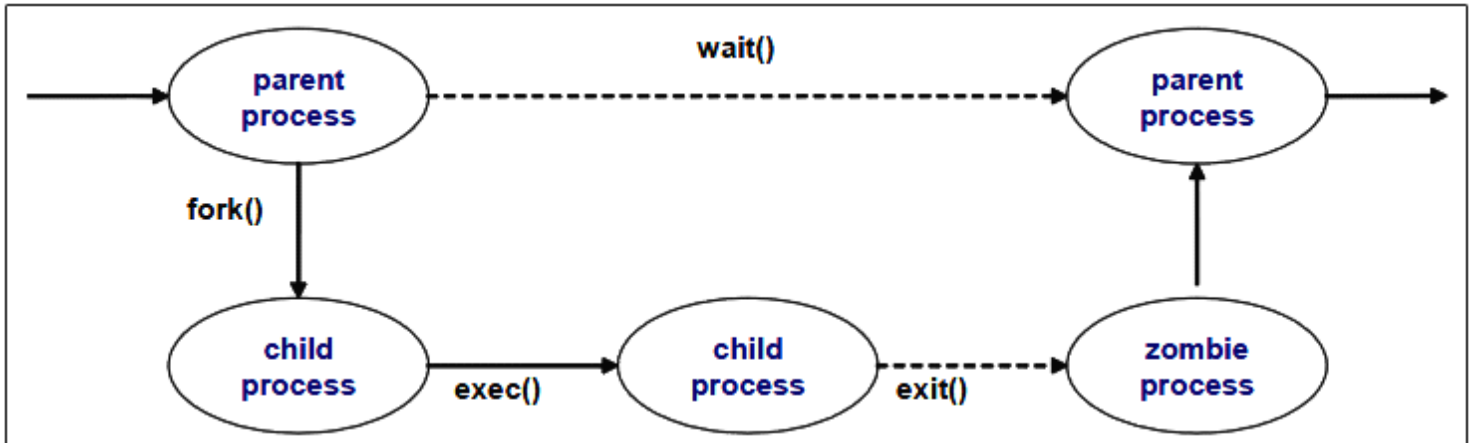


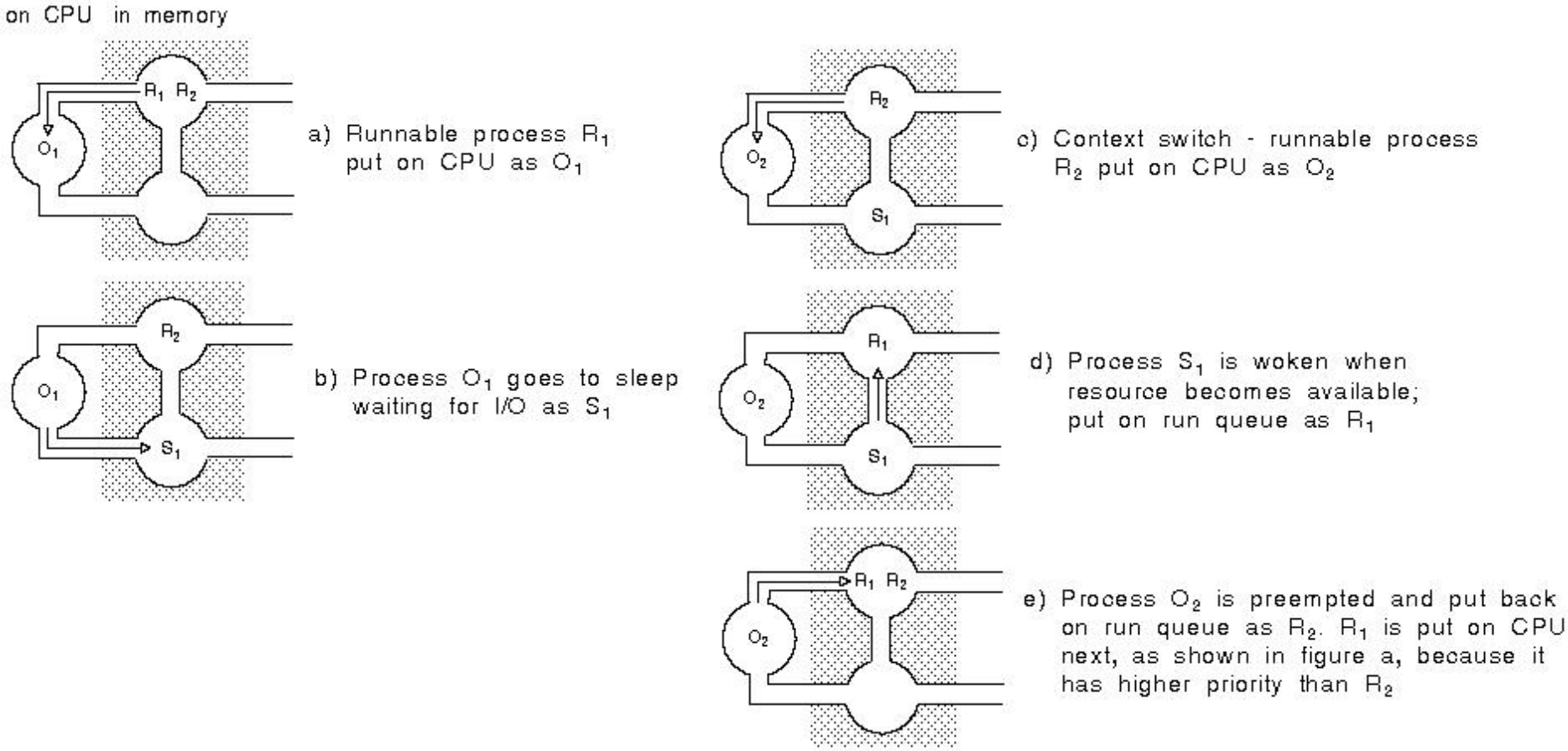
Figure 4.18 Linux Process/Thread Model

基础知识——僵尸进程



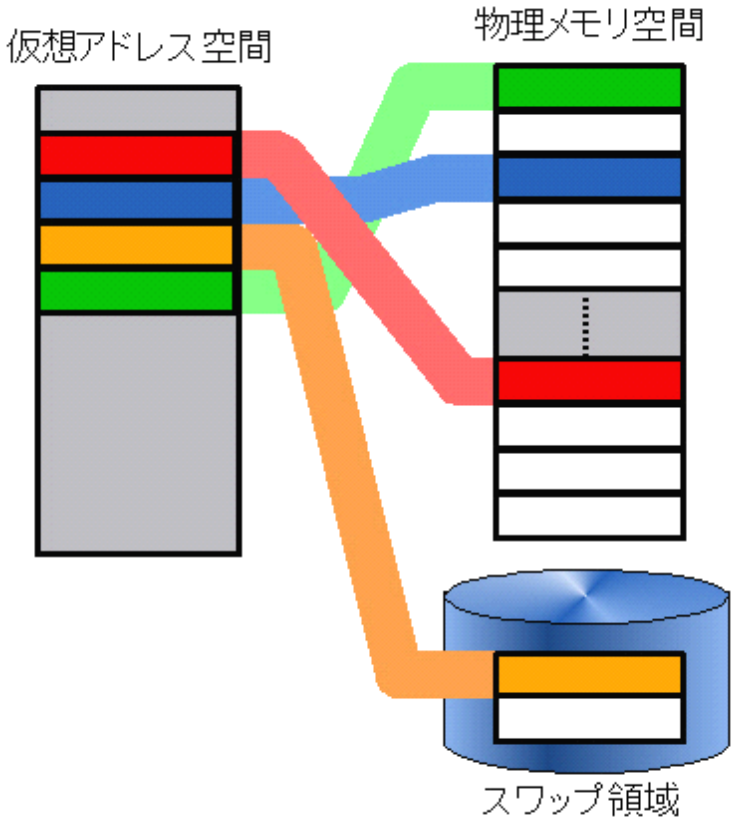
- 进程已死亡，但父进程没有收尸，该进程就成僵尸进程
- 僵尸进程不打开任何文件，几乎不占内存，但是占据进程表的资源，进程表记录pid、进程状态、CPU时间等
- 僵尸状态是每个子进程结束时必经的状态
- 系统监控中出现大量僵尸进程，应检查其父进程代码

基础知识——上下文切换



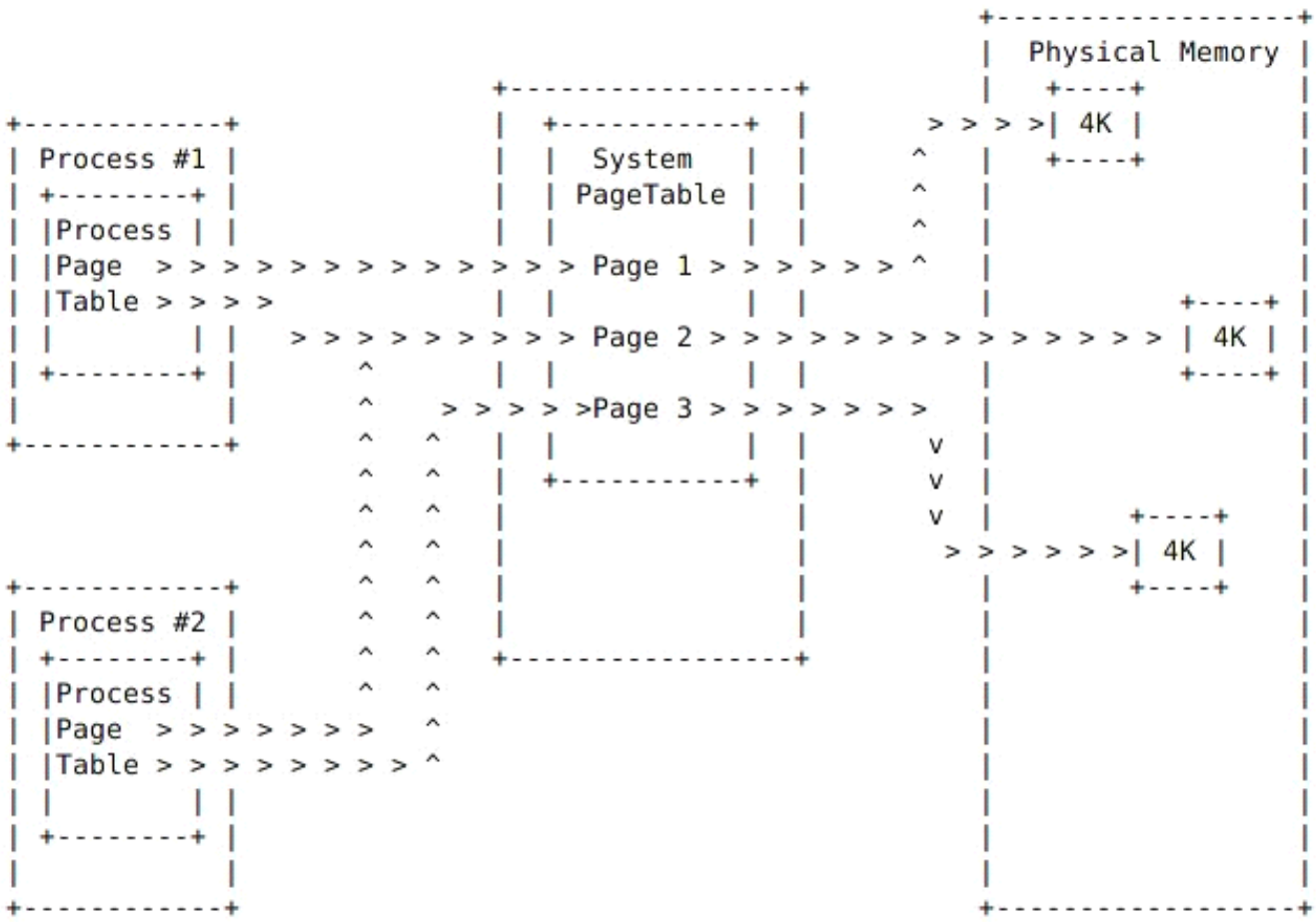
- 上下文是指进程在CPU的执行环境，例如寄存器状态
- 进程挂起时，保存其上下文，再次运行时恢复

基础知识——虚拟内存与页面



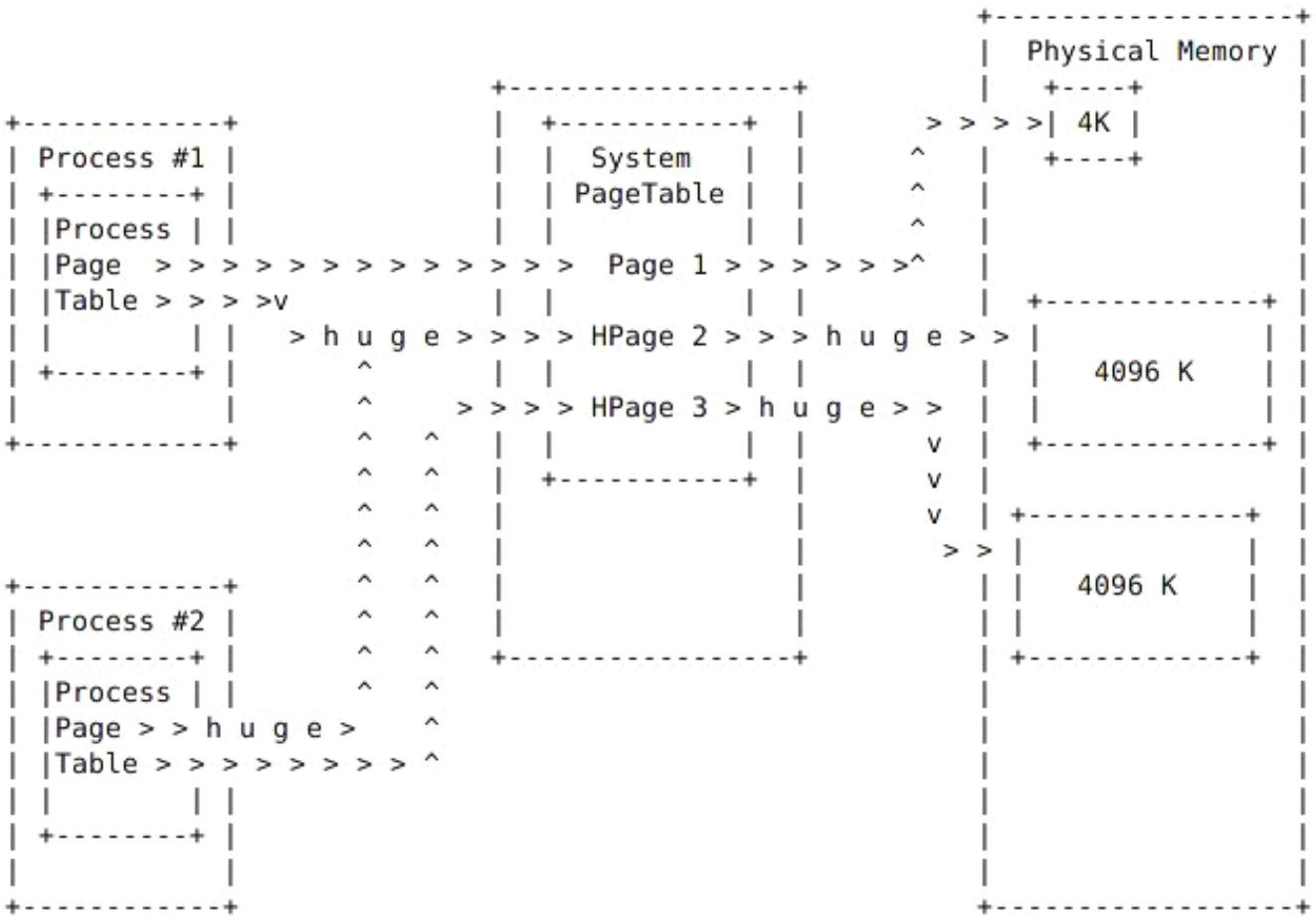
- 每个进程都有其地址空间，地址空间在逻辑上是连续的
- 物理上可以被分为多个RAM碎片，也可以有部分暂存在磁盘，使得进程可用的内存比实际内存要多
- 操作系统把虚拟内存分成一个个页面来管理，Linux的页面大小通常为4K
- 当要访问的逻辑页面不在物理内存时，产生缺页中断
- 内存不足时，淘汰旧页面，换入新页面，页面交换有paging和swapping之分

基础知识——常规页面



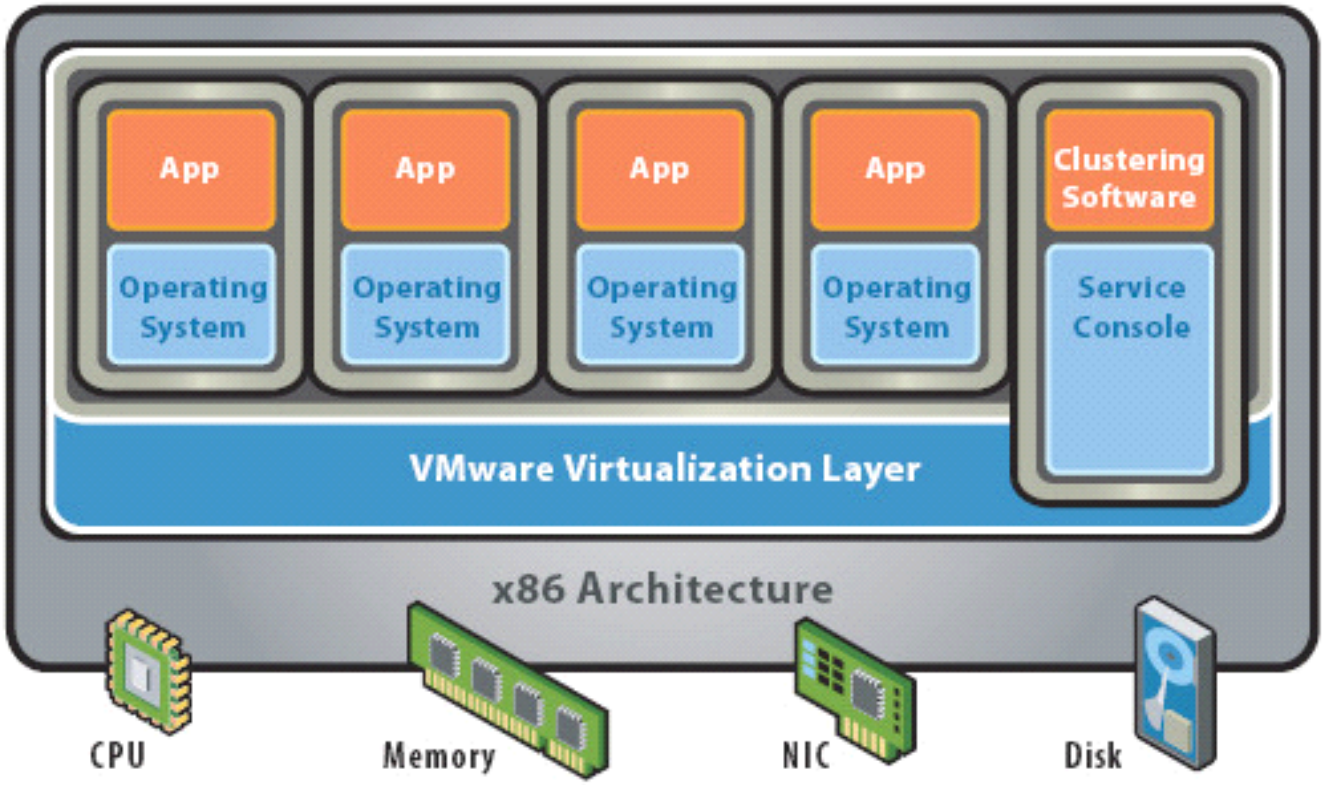
- 一些大型应用，例如关系数据库等，如仍然使用4K大小的页面，页表必然很大，影响效率

基础知识——巨页



- 使用巨页能够以较小的页表映射较多的内存

基础知识——虚拟化技术



- Hypervisor：运行在物理服务器和操作系统之间的中间软件层，可允许多个操作系统共享硬件
- 991原则：90%的服务器在90%的时间，CPU使用率低于10%

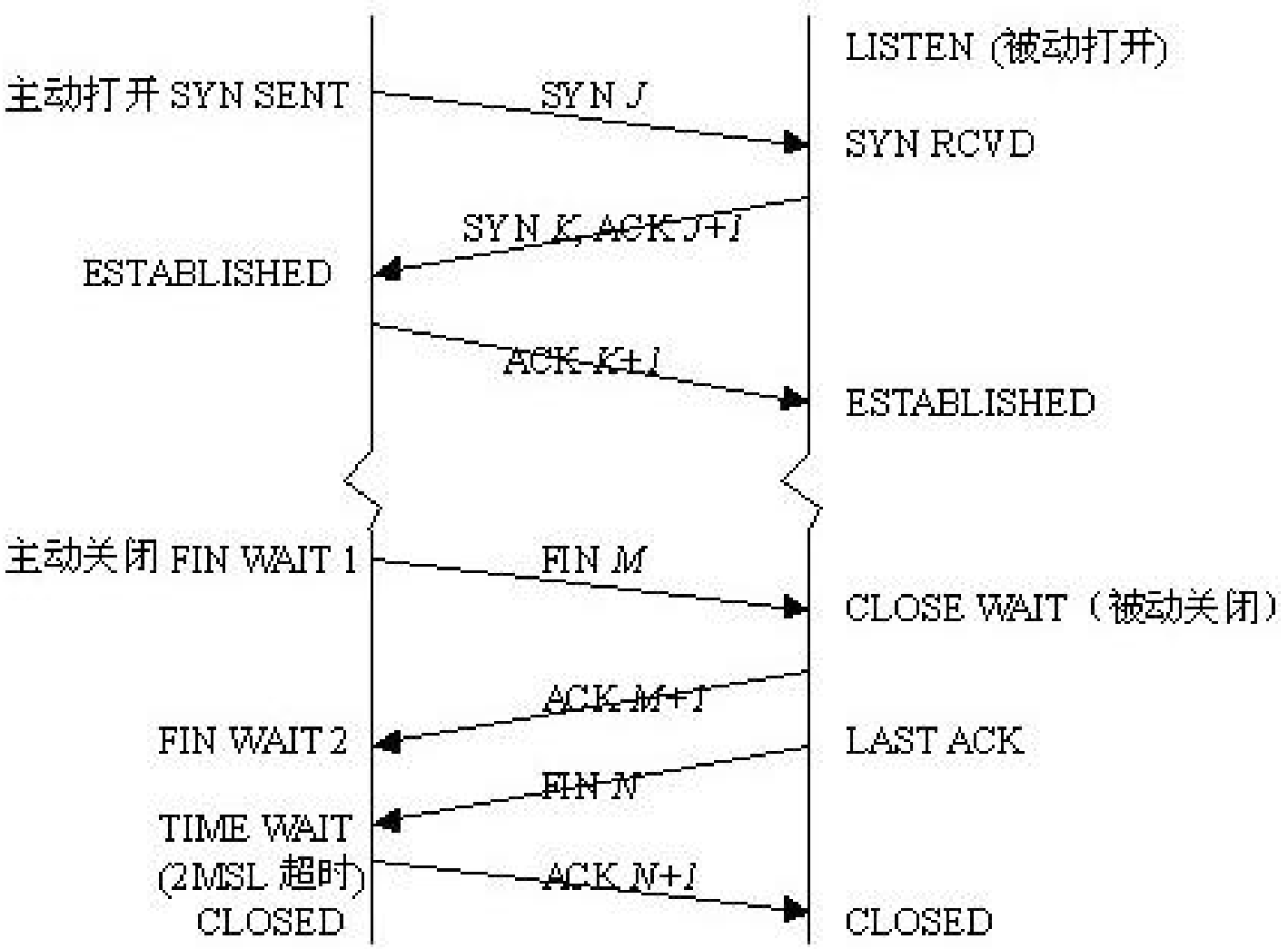
基础知识——tcp连接状态

- 连接建立时的tcp状态变迁
- 客户端：SYN_SENT、ESTABLISHED
- 服务端：LISTEN、SYN_RECV、ESTABLISHED

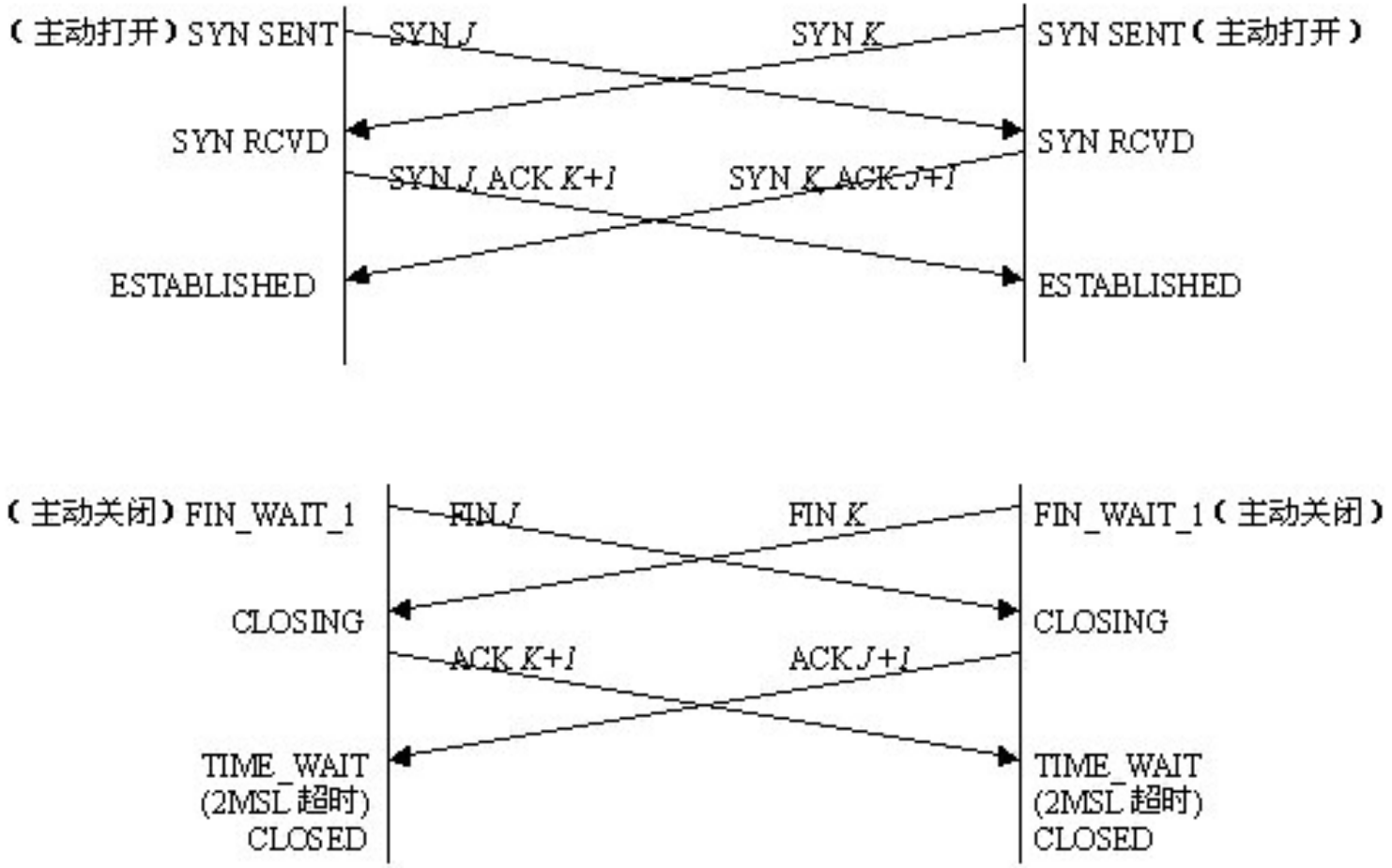
- 连接关闭时的tcp状态变迁
- 主动方：ESTABLISHED、FIN_WAIT1、FIN_WAIT2、TIME_WAIT、CLOSED
- 被动方：ESTABLISHED、CLOSE_WAIT、LAST_ACK、CLOSED

- 一个特殊的tcp状态
- 连接双方同时关闭：CLOSING

基础知识——三次握手与四次握手



基础知识——同时连接与同时关闭连接



基础知识小结

UC优视

- 核心态与用户态
- 系统调用
- 进程组与会话
- 进程的几种状态、僵尸进程、上下文切换
- 虚拟内存、缺页中断、页面交换、巨页
- tcp连接状态
- 虚拟化技术

- Q.为什么从这些概念开始？
- A.因为这些对于理解命令的输出大有裨益

内容

1 培训目标

2 基础知识

3 常用监控命令

4 在实战中综合运用

Linux top命令——汇总区域

```
top - 11:05:37 up  1:34,  7 users,  load average: 0.00, 0.00, 0.00
Tasks: 272 total,   1 running, 270 sleeping,   0 stopped,   1 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni, 99.9%id,  0.1%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  24676764k total,   826012k used, 23850752k free,   183000k buffers
Swap:        0k total,        0k used,        0k free,   287472k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5434 dengwl    16   0 96444 3436 2244 S  0.3  0.0   0:02.65 vim
 8753 liangry   15   0 12872 1252  828 R  0.3  0.0   0:00.38 top
    1 root      15   0 10348  696  584 S  0.0  0.0   0:02.42 init
```

- 汇总区域显示五个方面系统性能信息
- 负载：时间、登录用户数、系统平均负载
- 进程：运行、睡眠、停止、僵尸
- CPU：用户态、核心态、NICE、空闲、等待IO、中断等
- 内存：总量、已用、空闲（系统角度的）、缓冲、缓存
- 交换分区：总量、已用、空闲

Linux top命令——任务区域

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 9001 liangry   15   0 12872 1248  832 R  0.3  0.0   0:01.05 top
 8702 liangry   15   0 90120 1756 1008 S  0.0  0.0   0:00.18 sshd: liangry@pts/5
 8703 liangry   15   0 66092 1552 1192 S  0.0  0.0   0:00.02 -bash
 8813 liangry   15   0 90120 1748 1004 S  0.0  0.0   0:00.15 sshd: liangry@pts/7
 8814 liangry   15   0 66092 1548 1196 S  0.0  0.0   0:00.02 -bash
```

- 任务区域默认显示：进程ID、有效用户、进程优先级、NICE值、进程使用的虚拟内存、物理内存和共享内存、进程状态、CPU占用率、内存占用率、累计CPU时间、进程命令行信息
- 使用交互命令f可选择更多

列名	描述	列名	描述
PPID	父进程ID	CODE	代码段大小
RUSER	实际用户	DATA	数据段+栈大小
TTY	终端名	nFLT	缺页中断次数
P或者#C	进程调度的CPU	nDRT	脏页数
SWAP	进程使用的交换分区大小	……	……

Linux top命令——交互命令(1)

类型	命令	描述	备注
汇总区显示	数字1	CPU模式切换	
	字母l	负载信息开关	
	m	内存信息开关	
	t	进程信息开关	
色彩	Z	色彩选择	
	x	排序字段高亮	前置命令Z
	y	运行状态高亮	前置命令Z
	z	色彩显示开关	前置命令Z
任务区字段排列与排序	F或者O	排序字段选择	
	o	字段顺序选择	
	R	倒序、顺序切换	
	<和>	排序字段选择	前置命令Z、x

Linux top命令——交互命令(2)

类型	命令	描述	备注
任务区显示	c	命令行信息切换	
	f	显示字段选择	
	H	线程、进程显示切换	
	n或者#	设置显示的任务数	
	u	设置显示某个有效用户的任务	两者究竟有何区别？
	U	设置显示某个实际用户的任务	
其它	W	保存设置	
	空格或回车	手动刷新	

```
  PID USER      RUSER     PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8703 liangry   liangry   16   0 66092 1540 1184 S  0.0  0.0   0:00.01 -bash
 8843 root      liangry   16   0 53212 1292 1004 S  0.0  0.0   0:00.00 passwd
```

- 对于设置了粘着位的程序，有效用户与实际用户的区别

Linux iostat命令——主要命令行参数

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.02    0.04    0.00   99.92

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               3.84        57.17         9.58     401305      67260
sdb               2.09         9.04         1.22      63441       8572
```

```
Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu
-sz    await r_await w_await  svctm  %util
sda               0.00     0.00 1800.00    0.00  7200.00     0.00     8.00     0
.93     0.52    0.52    0.00    0.52  93.20
```

- iostat -c：见top命令的CPU信息
- iostat -d：吞吐率、每秒读写、总的读写
- iostat -x：每秒读写次数、平均IO队列长度、平均每次IO操作总耗时、平均每次IO操作的实际执行时间、IO使用率
- 一个隐藏的等式：%util = (r/s + w/s) * svctm / 10

Linux iostat命令——案例分析(1)

- 遭遇IO瓶颈

	r/s	w/s	avgqu-sz	await	svctm	%util
……	71.60	125.00	68.68	356.60	4.75	93.42
……	88.40	74.20	38.25	236.09	5.45	88.54
……	44.60	124.00	84.49	510.51	5.75	96.98

- (71.60+125.00)*4.75 = 933.85(ms)
- (88.40+74.20)*5.45 = 886.17(ms)
- (44.60+124.00)*5.75 = 969.45(ms)
- 那么，IO瓶颈有什么症状？

- %util很高
- await远远大于svctm
- avgqu-sz比较大

Linux iostat命令——案例分析(2)

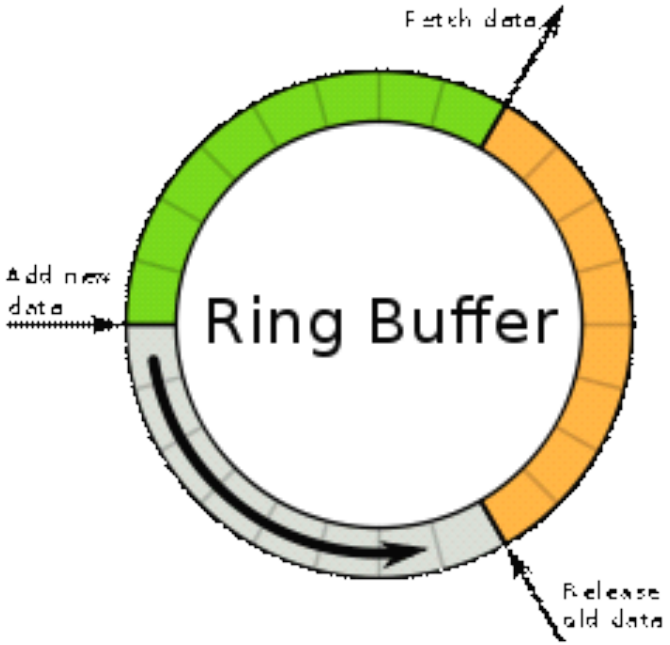
- 只是IO忙碌

	r/s	w/s	avgqu-sz	await	svctm	%util
……	194.40	1.20	1.04	5.32	5.05	98.70
……	174.00	1.80	1.04	5.90	5.63	99.00
……	217.20	1.20	1.01	4.62	4.50	98.28

- (194.40+1.20)*5.05 = 987.78(ms)
- (174.00+1.80)*5.63 = 989.75(ms)
- (217.20+1.20)*4.50 = 982.80(ms)
- 为什么不算IO瓶颈？

- %util还是很高，比较容易迷惑人
- await接近svctm，略大一点点
- avgqu-sz不大

Linux dmesg命令



- 打印内核“环缓冲区”的信息
- 内核“环缓冲区”保存Linux开机信息
- 也记录某些程序的内存错误（segfault）

```
[liangry@platform4 nginx-0.7.62]$ dmesg | tail -1
cscope[4749]: segfault at 0000000000000302 rip 0000002a95591aff rsp 0000007fbfff
ef48 error 4
```

Linux lsof命令(1)

- 查看文件系统阻塞，解决umount busy问题

```
[root@platform4 sbin]# /usr/sbin/lsof /boot
COMMAND  PID    USER   FD   TYPE DEVICE SIZE NODE NAME
bash    4192 liangry  cwd    DIR    8,1 4096    2 /boot
```

- 查看监听端口被哪个进程占用

```
[root@platform4 sbin]# /usr/sbin/lsof -i :8080
COMMAND  PID    USER   FD   TYPE DEVICE SIZE NODE NAME
nginx   4762 liangry   10u  IPv4  11712       TCP *:webcache (LISTEN)
nginx   4838 liangry   10u  IPv4  11712       TCP *:webcache (LISTEN)
```

- 查看用户打开哪些文件

```
[root@platform4 sbin]# /usr/sbin/lsof -u liangry
COMMAND  PID    USER   FD   TYPE             DEVICE      SIZE    NODE NAME
sshd    4585 liangry  cwd    DIR                8,2      4096       2 /
sshd    4585 liangry  rtd    DIR                8,2      4096       2 /
sshd    4585 liangry  txt    REG                8,6    346376  737424 /usr/
sshd    4585 liangry  mem    REG                8,2    115168  414362 /lib6
```

Linux lsof命令(2)

- 查看进程打开哪些文件

```
[root@platform4 sbin]# /usr/sbin/lsof -p 4838
COMMAND  PID    USER   FD   TYPE             DEVICE      SIZE    NODE NAME
nginx   4838 liangry  cwd    DIR                8,8      4096 1705022 /home/lia
nginx   4838 liangry  rtd    DIR                8,2      4096       2 /
nginx   4838 liangry  txt    REG                8,8   2782180 1703937 /home/lia
```

- 查看远程已打开的网络连接

```
[root@platform4 sbin]# /usr/sbin/lsof -i @192.168.34.128
COMMAND  PID    USER   FD   TYPE DEVICE SIZE NODE NAME
sshd    4583    root    3u  IPv6  11076       TCP platform4:9922->192.168.34.128
sshd    4585 liangry    3u  IPv6  11076       TCP platform4:9922->192.168.34.128
```

Linux netstat命令

- 常规用法

netstat参数	描述	执行格式
-r	显示路由表	netstat -r
-i	显示网络接口	netstat -i
-a	显示所有socket	netstat -a或netstat -an
-n	显示IP	与-a等参数联合使用

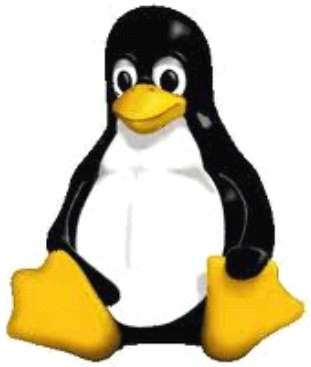
- 小技巧
- 汇总统计tcp连接状态

```
[liangry@platform1 ~]$ netstat | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a
, S[a]}'
TIME_WAIT 18
ESTABLISHED 91
```

- 查看端口被哪个进程占用

```
[root@platform4 sbin]# netstat -pan | grep 8080
tcp        0      0 0.0.0.0:8080                0.0.0.0:*                   LIST
EN      4762/nginx
```

Linux ps命令



	BSD	Linux
基本用法	ps aux	ps -ef
显示PGID和SID	ps auxj	ps -efj
显示进程派生树	ps auxf	ps -efH

- 进程状态：D、R(unning)、S(leep)、T、Z(ombie)
- BSD风格附加状态：<(高优先级)、N(ice)、s(领头进程)、l(多线程)、+(前台进程)

Linux vmstat命令

- 内存不足以致页面交换频繁，系统性能下降
- 从下面的截图可以看出哪些性能指标？

```
[liangry@platform4 sbin]$ vmstat 5
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0 832608  42212  92496    0    0     3     2  506   10  0  0 100  0
 1  0      0 280280  42244  92464    0    0     2    17 1017   26 19  7 73  0
 1  4 161212  15896    460   4112  154 33070   439 33102 1165  155 19 14 16 51
 0  6 352700  15872    484   6400  166 37466   609 37470 1112  134  7  6  6 81
 0  5 550856  15872    476   5584    0 39634     0 39635 1091  143  8  7  0 85
 0  5 754200  15872    464   3612   26 40668    34 40673 1095  152  8  7  3 82
 0  6 938384  17856    444   3984   45 36838   134 36841 1085  133  6  6  2 86
 0  6 1139636  15864    464   4096   26 40262    27 40270 1102  160  8  8  0 85
 0  0 1200692  25208    444   3784   96 12238   145 12242 1061   67  2  4 58 36
 0  0 1200692  25416    452   3844    0    0     0     3 1013   19  0  0 100  0
 0  0 1200692  25680    452   3844    0    0     0     0 1009   14  0  0 100  0
```

- free memory急剧减少，回收buffer和cache也无济于事，于是大量使用交换分区（swpd），页面交换（swap）频繁，读写磁盘数量（io）增多，缺页中断（in）增多，上下文切换（cs）次数增多，等待IO的进程数（b）增多，大量CPU时间用于等待IO（wa）

Linux sar命令——实例(1)

- 监控磁盘IO，与iostat类似

```
[liangry@platform4 sbin]$ sar -bd 5
Linux 2.6.9-89.35.1.ELsmp (platform4)   07/11/11        _x86_64_        (2 CPU)

17:09:00          tps      rtps      wtps   bread/s   bwrtn/s
17:09:05        81.27     19.29     61.99    909.36  68858.43

17:09:00          DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
17:09:05       dev8-0     81.27    909.36  68858.43    858.43    110.14    786.46     11.48     93.33
17:09:05      dev8-16      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

- 监控页面交换，paging与swapping
- 如果页面回收率（%vmeff）接近100，表示几乎所有被列入回收列表的页面都能够被正常回收
- 如果%vmeff太低，又不是0.00，表示内存有瓶颈

```
[liangry@platform4 sbin]$ sar -WB 5
Linux 2.6.9-89.35.1.ELsmp (platform4)   07/11/11        _x86_64_        (2 CPU)

17:17:48     pswpin/s pswpout/s
17:17:53        32.78   6558.56

17:17:48     pgpgin/s pgpgout/s   fault/s  majflt/s  pgfree/s pgscank/s pgscand/s pgsteal/s    %vmeff
17:17:53       691.71  26262.25  17304.97     12.89   4646.41   5822.10   6016.57   4631.86     39.12
```

Linux sar命令——实例(2)

- 监控内存、交换分区、巨页页面

```
[liangry@platform4 sbin]$ sar -RrSH 5
Linux 2.6.9-89.35.1.ELsmp (platform4)   07/11/11        _x86_64_        (2 CPU)

17:28:28      frmpg/s   bufpg/s   campg/s
17:28:33     -3430.66    -50.91   -106.39

17:28:28    kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact
17:28:33        16200   1008660     98.42       448      5504   2128012     64.37    490464    483396

17:28:28    kbswpfree kbswpused  %swpused  kbswpcad   %swpcad
17:28:33      2119864    161356      7.07     55520     34.41

17:28:28    kbhugfree kbhugused  %hugused
17:28:33            0         0      0.00
```

- sar -R：空闲内存页面、缓冲页面和缓存页面的变化趋势
- sar -r：当前内存的使用信息，与free命令类似
- sar -S：当前交换分区的使用信息，与free命令类似
- sar -H：当前巨页页面的信息

Linux sar命令——实例(3)

- 监控CPU、进程创建与上下文切换、运行队列与负载

```
[liangry@platform4 sbin]$ sar -uwq 5
Linux 2.6.9-89.35.1.ELsmp (platform4)   07/11/11        _x86_64_        (2 CPU)

17:37:37        CPU     %user     %nice   %system   %iowait    %steal     %idle
17:37:42        all      5.79      0.00      5.40     88.81      0.00      0.00

17:37:37       proc/s   cswch/s
17:37:42         0.00    101.17

17:37:37      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15   blocked
17:37:42            1        82      1.37      0.42      0.31         3
```

- proc/s：平均每秒创建的进程数
- cswch/s：平均每秒的上下文切换次数
- runq-sz：等待运行时间片的进程数
- plist-sz：进程列表的总数
- ldavg-x：过去一段时间内的系统平均负载，即进程状态为R或D的加权平均数
- blocked：当前被IO阻塞的进程数

内容

1 培训目标

2 基础知识

3 常用监控命令

4 在实战中综合运用

在实战中综合运用(1)

- 首先，使用w查看系统负载

$ w

10:46:36 up 85 days, 14:17, 4 users, load average: 6.52, 5.57, 5.55

分析：负载比较高，系统资源出现瓶颈

- 第二，使用vmstat查看系统大致状况

$ vmstat 2

```
procs -----------memory---------- ---swap-- -----io---- --system--  -----cpu------
r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs    us sy id wa st
0  4  36212 524784 126016 4780220    0    0    33    21    0    0    6  7 80  7  0
2  0  36212 468396 126664 4800532    0    0   602 15382 13900 18410  9 14 69  8  0
2  3  36212 441572 127064 4792600    0    0  2024 56110 13754 14710  8 11 68 14  0
0  0  36212 477532 127572 4777576    0    0  1208 20356 13661 8726   5  9 70 17  0
0  1  36212 509236 128116 4744336    0    0  1880 11172 13646 11622  7  9 80  4  0
6  1  36212 508344 128688 4722108    0    0  2290 23354 13815 10977  7 10 73 10  0
4  0  36212 459968 129264 4766660    0    0  1404    64 13357 16221  9 12 77  3  0
1  2  36212 462180 129712 4752460    0    0   486 40270 13674 14392  8 11 70 10  0
```

分析：交换分区无变化、也无页面交换，排除内存不足；CPU的idle值维持在70%以上，基本可排除CPU的问题；但是，每秒写的块数（bo）很大，CPU的IO wait（wa）也不低，IO方面可能存在瓶颈

在实战中综合运用(2)

- 第三，使用iostat分析IO瓶颈

$ iostat -d -x 1

```
Device:         rrqm/s   wrqm/s   r/s   w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.00  8320.00  0.00 470.00     0.00 97552.00   207.56   135.86  304.25   2.13 100.10
sda1              0.00     0.00  0.00   0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sda2              0.00     0.00  0.00   0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
...
sda8              0.00  8320.00  0.00 470.00     0.00 97552.00   207.56   135.86  304.25   2.13 100.10
sdb              41.00    10.00 24.00 120.00  6328.00  1040.00    51.17     5.75   39.92   2.00  28.80
sdb1             41.00    10.00 24.00 120.00  6328.00  1040.00    51.17     5.75   39.92   2.00  28.80
```

分析：IO集中在sda8，%util很高，await远远大于svctm，avgqu-sz也很大，满足IO瓶颈的全部条件

- 第四，使用df查看磁盘空闲块和inode

$ df -h /dev/sda8

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda8             114G  1.5G  106G   2% /home
```

$ df -i /dev/sda8

```
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/sda8               30M     19K     30M    1% /home
```

分析：均有空闲，排除资源接近耗尽，此外，业务日志写在sdb1分区，也可排除写入大量业务日志带来的压力

在实战中综合运用(3)

- 第五，使用lsof查看什么进程操作/home目录下的文件

$ lsof -n | grep home

```
nginx      6010       cws  286u      REG        8,8 16760832   11206661
    /home/cws/cwsserver/fastcgi_temp/0000002200 (deleted)
nginx      6010       cws  687u      REG        8,8  7332864   11206673
    /home/cws/cwsserver/fastcgi_temp/0000002397 (deleted)
nginx      6011       cws  473u      REG        8,8 16760832   11206671
    /home/cws/cwsserver/fastcgi_temp/0000001976 (deleted)
...
```

分析：没有发现其它异常，但有大量的nginx fastcgi信息，看上去与此有关

- 第六，排查、修改nginx的fastcgi设置

解决：nginx.conf中没有fastcgi的设置，即使用默认参数，fastcgi缓冲区大小为36k，对于部分response来说太小，把它适当改大，问题解决。使用vmstat、iostat等命令再查看皆恢复正常

Thank You!

uc.cn