（5） Linux方面：IO模型，linux常见命令（grep，awk，sed），文件系统，

4Linux基础  
Linux这一块如果不太熟，可以直接说，但是因为开发岗位一般都是在Linux平台下的，所以几个基础问题还是要会的。  
（1） Linux的I/O模型介绍以及同步异步阻塞非阻塞的区别（超级重要）  
（2） 文件系统的理解（EXT4，XFS，BTRFS）  
（3） 文件处理grep,awk,sed这三个命令必知必会  
（4） IO复用的三种方法（select,poll,epoll）深入理解，包括三者区别，内部原理实现？  
（5） Epoll的ET模式和LT模式（ET的非阻塞）

（6） 查询进程占用CPU的命令（注意要了解到used，buf，cache代表意义）

我们使用top命令来查看CPU使用状况。  
top不会产生输出，屏幕内容保持不变。它刷新屏幕以显示新信息。因此，如果您只执行top并保持屏幕一直开启，则屏幕始终显示最新信息。退出top的命令为q,或者按下Ctrl-C.  
top - 17:03:45 up 58 days,  4:01,  1 user,  load average: 0.00, 0.02, 0.00   
Tasks: 172 total,  1 running, 171 sleeping,  0 stopped,  0 zombie  
Cpu(s):  0.2% us,  0.1% sy,  0.0% ni, 99.7% id,  0.0% wa,  0.0% hi,  0.0% si  
Mem:  4037036k total,  4007280k used,    29756k free,    93384k buffers  
Swap:  8385888k total,    71536k used,  8314352k free,  3068240k cached  
  
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND                                                                
27167 oracle    16  0 2011m 490m 483m S    1 12.4  0:46.93 oracle                                                                  
27175 oracle    15  0 2011m 517m 510m S    1 13.1  0:49.78 oracle                                                                  
5003 oracle    15  0 2021m  33m  29m S    0  0.9  3:56.10 oracle                                                                  
    1 root      16  0  4756  552  460 S    0  0.0  0:09.31 init                                                                    
    2 root      RT  0    0    0    0 S    0  0.0  0:00.47 migration/0                                                              
  ...............................  
第一行（top）：  
top - 17:03:45 up 58 days,  4:01,  1 user,  load average: 0.00, 0.02, 0.00   
　　　　“17:03:45”为系统当前时刻；  
　　　　“58 days,  4:01”为系统启动后到现在的运作时间；  
　　　　“1 user”为当前登录到系统的用户，更确切的说是登录到用户的终端数--同一个用户同一时间对系统多个终端的连接将被视为多个用户连接到系统，这里的用户数也将表现为终端的数目；  
　　　　“load average”为当前系统负载的平均值，后面的三个值分别为1分钟前、5分钟前、15分钟前进程的平均数，一般的可以认为这个数值超过CPU数目时，CPU将比较吃力的负载当前系统所包含的进程；  
第二行（Tasks）：  
　　　　“172 total”为当前系统进程总数；  
　　　　“1 running”为当前运行中的进程数；  
　　　　“171 sleeping”为当前处于等待状态中的进程数；  
　　　　“0 stoped”为被停止的系统进程数；  
　　　　“0 zombie”为僵死的进程数；  
第三行（Cpus）：  
　　显示CPU利用率的详细信息，如果有多个CPU，屏幕将在每行显示一个CPU的信息。  
第四行（Mem）：  
　　显示可用的和已利用的内存  
第五行（Swap）：  
　　　　表示类别同第四行（Mem），但此处反映着交换分区（Swap）的使用情况。通常，交换分区（Swap）被频繁使用的情况，将被视作物理内存不足而造成的。  
　　其余的显示内容以表格格式显示进程。下面对各列进行解释：  
列描述  
PID            进程的进程ID  
USER          运行该进程的用户  
PRI          进程的优先级  
NI            nice值：该值越高，任务的优先级越低  
SIZE          该进程使用的内存（代码+数据+堆栈）  
RSS          该进程使用的物理内存  
SHARE        该进程使用的共享内存  
STAT  该进程的状态，用代码显示。一些主要的状态代码包括：  
R— 正在运行  
S— 正在休眠  
Z— 迟滞  
T— 已停止  
您还会看到第二个和第三个字符，它们表示：  
W— 已换出的进程  
N— 正nice值  
%CPU          该进程使用的CPU百分比  
%MEM          该进程使用的内存百分比  
TIME          该进程使用的总CPU时间  
CPU          如果这是一个多处理器系统，该列指明正在其上运行进程的CPU的ID。  
COMMAND        该进程发出的命令  
top运行中可以通过top的内部命令对进程的显示方式进行控制。内部命令如下：  
　　s -改变画面更新频率  
　　l -关闭或开启第一部分第一行top信息的表示  
　　t -关闭或开启第一部分第二行Tasks和第三行Cpus信息的表示  
　　m -关闭或开启第一部分第四行Mem和第五行Swap信息的表示  
　　N -以PID的大小的顺序排列表示进程列表  
    P -以CPU占用率大小的顺序排列进程列表    
  M -以内存占用率大小的顺序排列进程列表  
　　h -显示帮助  
　　n -设置在进程列表所显示进程的数量  
　　q -退出top

Linux中的ps命令是Process Status的缩写。ps命令用来列出系统中当前运行的那些进程。ps命令列出的是当前那些进程的快照，就是执行ps命令的那个时刻的那些进程，如果想要动态的显示进程信息，就可以使用top命令。

要对进程进行监测和控制，首先必须要了解当前进程的情况，也就是需要查看当前进程，而 ps 命令就是最基本同时也是非常强大的进程查看命令。使用该命令可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等等。总之大部分信息都是可以通过执行该命令得到的。

ps 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 top 工具。

kill 命令用于杀死进程。

**linux上进程有5种状态:**

1. 运行(正在运行或在运行队列中等待)

2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)

3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)

4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)

5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行运行)

**ps工具标识进程的5种状态码:**

D 不可中断 uninterruptible sleep (usually IO)

R 运行 runnable (on run queue)

S 中断 sleeping

T 停止 traced or stopped

Z 僵死 a defunct (”zombie”) process

**1．命令格式：**

ps[参数]

**2．命令功能：**

用来显示当前进程的状态

**3．命令参数：**

a  显示所有进程

-a 显示同一终端下的所有程序

-A 显示所有进程

c  显示进程的真实名称

-N 反向选择

-e 等于“-A”

e  显示环境变量

f  显示程序间的关系

-H 显示树状结构

r  显示当前终端的进程

T  显示当前终端的所有程序

u  指定用户的所有进程

-au 显示较详细的资讯

-aux 显示所有包含其他使用者的行程

-C<命令> 列出指定命令的状况

--lines<行数> 每页显示的行数

--width<字符数> 每页显示的字符数

--help 显示帮助信息

--version 显示版本显示

**4．使用实例：**

**实例1：显示所有进程信息**

**命令：**

ps -A

**输出：**

[root@localhost test6]# ps -A

  PID TTY          TIME CMD

    1 ?        00:00:00 init

    2 ?        00:00:01 migration/0

    3 ?        00:00:00 ksoftirqd/0

    4 ?        00:00:01 migration/1

    5 ?        00:00:00 ksoftirqd/1

    6 ?        00:29:57 events/0

    7 ?        00:00:00 events/1

    8 ?        00:00:00 khelper

   49 ?        00:00:00 kthread

   54 ?        00:00:00 kblockd/0

   55 ?        00:00:00 kblockd/1

   56 ?        00:00:00 kacpid

  217 ?        00:00:00 cqueue/0

  ……省略部分结果

**说明：**

**实例2：显示指定用户信息**

**命令：**

ps -u root

**输出：**

[root@localhost test6]# ps -u root

  PID TTY          TIME CMD

    1 ?        00:00:00 init

    2 ?        00:00:01 migration/0

    3 ?        00:00:00 ksoftirqd/0

    4 ?        00:00:01 migration/1

    5 ?        00:00:00 ksoftirqd/1

    6 ?        00:29:57 events/0

    7 ?        00:00:00 events/1

    8 ?        00:00:00 khelper

   49 ?        00:00:00 kthread

   54 ?        00:00:00 kblockd/0

   55 ?        00:00:00 kblockd/1

   56 ?        00:00:00 kacpid

    ……省略部分结果

**说明：**

**实例3：显示所有进程信息，连同命令行**

**命令：**

ps -ef

**输出：**

[root@localhost test6]# ps -ef

UID        PID  PPID  C STIME TTY          TIME CMD

root         1     0  0 Nov02 ?        00:00:00 init [3]

root         2     1  0 Nov02 ?        00:00:01 [migration/0]

root         3     1  0 Nov02 ?        00:00:00 [ksoftirqd/0]

root         4     1  0 Nov02 ?        00:00:01 [migration/1]

root         5     1  0 Nov02 ?        00:00:00 [ksoftirqd/1]

root         6     1  0 Nov02 ?        00:29:57 [events/0]

root         7     1  0 Nov02 ?        00:00:00 [events/1]

root         8     1  0 Nov02 ?        00:00:00 [khelper]

root        49     1  0 Nov02 ?        00:00:00 [kthread]

root        54    49  0 Nov02 ?        00:00:00 [kblockd/0]

root        55    49  0 Nov02 ?        00:00:00 [kblockd/1]

root        56    49  0 Nov02 ?        00:00:00 [kacpid]

……省略部分结果

**说明：**

**实例4：** **ps 与grep 常用组合用法，查找特定进程**

**命令：**

ps -ef|grep ssh

**输出：**

[root@localhost test6]# ps -ef|grep ssh

root      2720     1  0 Nov02 ?        00:00:00 /usr/sbin/sshd

root     17394  2720  0 14:58 ?        00:00:00 sshd: root@pts/0

root     17465 17398  0 15:57 pts/0    00:00:00 grep ssh

**说明：**

**实例5：将目前属于您自己这次登入的 PID 与相关信息列示出来**

**命令：**

ps -l

**输出：**

[root@localhost test6]# ps -l

F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD

4 S     0 17398 17394  0  75   0 - 16543 wait   pts/0    00:00:00 bash

4 R     0 17469 17398  0  77   0 - 15877 -      pts/0    00:00:00 ps

**说明：**

各相关信息的意义：

F 代表这个程序的旗标 (flag)， 4 代表使用者为 super user

S 代表这个程序的状态 (STAT)，关于各 STAT 的意义将在内文介绍

UID 程序被该 UID 所拥有

PID 就是这个程序的 ID ！

PPID 则是其上级父程序的ID

C CPU 使用的资源百分比

PRI 这个是 Priority (优先执行序) 的缩写，详细后面介绍

NI 这个是 Nice 值，在下一小节我们会持续介绍

ADDR 这个是 kernel function，指出该程序在内存的那个部分。如果是个 running的程序，一般就是 "-"

SZ 使用掉的内存大小

WCHAN 目前这个程序是否正在运作当中，若为 - 表示正在运作

TTY 登入者的终端机位置

TIME 使用掉的 CPU 时间。

CMD 所下达的指令为何

在预设的情况下， ps 仅会列出与目前所在的 bash shell 有关的 PID 而已，所以， 当我使用 ps -l 的时候，只有三个 PID。

**实例6：列出目前所有的正在内存当中的程序**

**命令：**

ps aux

**输出：**

[root@localhost test6]# ps aux

USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND

root         1  0.0  0.0  10368   676 ?        Ss   Nov02   0:00 init [3]

root         2  0.0  0.0      0     0 ?        S<   Nov02   0:01 [migration/0]

root         3  0.0  0.0      0     0 ?        SN   Nov02   0:00 [ksoftirqd/0]

root         4  0.0  0.0      0     0 ?        S<   Nov02   0:01 [migration/1]

root         5  0.0  0.0      0     0 ?        SN   Nov02   0:00 [ksoftirqd/1]

root         6  0.0  0.0      0     0 ?        S<   Nov02  29:57 [events/0]

root         7  0.0  0.0      0     0 ?        S<   Nov02   0:00 [events/1]

root         8  0.0  0.0      0     0 ?        S<   Nov02   0:00 [khelper]

root        49  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kthread]

root        54  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kblockd/0]

root        55  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kblockd/1]

root        56  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kacpid]

……省略部分结果

**说明：**

USER：该 process 属于那个使用者账号的

PID ：该 process 的号码

%CPU：该 process 使用掉的 CPU 资源百分比

%MEM：该 process 所占用的物理内存百分比

VSZ ：该 process 使用掉的虚拟内存量 (Kbytes)

RSS ：该 process 占用的固定的内存量 (Kbytes)

TTY ：该 process 是在那个终端机上面运作，若与终端机无关，则显示 ?，另外， tty1-tty6 是本机上面的登入者程序，若为 pts/0 等等的，则表示为由网络连接进主机的程序。

STAT：该程序目前的状态，主要的状态有

R ：该程序目前正在运作，或者是可被运作

S ：该程序目前正在睡眠当中 (可说是 idle 状态)，但可被某些讯号 (signal) 唤醒。

T ：该程序目前正在侦测或者是停止了

Z ：该程序应该已经终止，但是其父程序却无法正常的终止他，造成 zombie (疆尸) 程序的状态

START：该 process 被触发启动的时间

TIME ：该 process 实际使用 CPU 运作的时间

COMMAND：该程序的实际指令

**实例7：列出类似程序树的程序显示**

**命令：**

ps -axjf

**输出：**

[root@localhost test6]# ps -axjf

Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.7/FAQ

 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND

    0     1     1     1 ?           -1 Ss       0   0:00 init [3]

    1     2     1     1 ?           -1 S<       0   0:01 [migration/0]

    1     3     1     1 ?           -1 SN       0   0:00 [ksoftirqd/0]

    1     4     1     1 ?           -1 S<       0   0:01 [migration/1]

    1     5     1     1 ?           -1 SN       0   0:00 [ksoftirqd/1]

    1     6     1     1 ?           -1 S<       0  29:58 [events/0]

    1     7     1     1 ?           -1 S<       0   0:00 [events/1]

    1     8     1     1 ?           -1 S<       0   0:00 [khelper]

    1    49     1     1 ?           -1 S<       0   0:00 [kthread]

   49    54     1     1 ?           -1 S<       0   0:00  \\_ [kblockd/0]

   49    55     1     1 ?           -1 S<       0   0:00  \\_ [kblockd/1]

   49    56     1     1 ?           -1 S<       0   0:00  \\_ [kacpid]

**说明：**

**实例8：找出与 cron 与 syslog 这两个服务有关的 PID 号码**

**命令：**

**输出：**

[root@localhost test6]# ps aux | egrep '(cron|syslog)'

root      2682  0.0  0.0  83384  2000 ?        Sl   Nov02   0:00 /sbin/rsyslogd -i /var/run/syslogd.pid -c 5

root      2735  0.0  0.0  74812  1140 ?        Ss   Nov02   0:00 crond

root     17475  0.0  0.0  61180   832 pts/0    S+   16:27   0:00 egrep (cron|syslog)

[root@localhost test6]#

**说明：**

**其他实例：**

1. 可以用 | 管道和 more 连接起来分页查看

命令：

ps -aux |more

2. 把所有进程显示出来，并输出到ps001.txt文件

命令：

ps -aux > ps001.txt

3. 输出指定的字段

命令：

 ps -o pid,ppid,pgrp,session,tpgid,comm

输出：

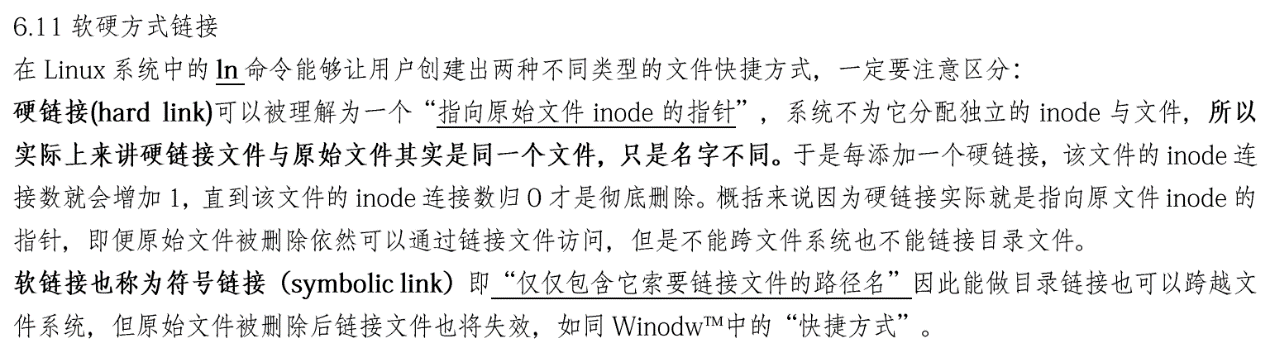
[root@localhost test6]# ps -o pid,ppid,pgrp,session,tpgid,comm

  PID  PPID  PGRP  SESS TPGID COMMAND

17398 17394 17398 17398 17478 bash

17478 17398 17478 17398 17478 ps

[root@localhost test6]#  
（7） linux的其他常见命令（kill，find，cp等等）  
（8） shell脚本用法  
（9） 硬连接和软连接的区别



为解决文件的共享使用，Linux 系统引入了两种链接：硬链接 (hard link) 与软链接（又称符号链接，即 soft link 或 symbolic link）。链接为 Linux 系统解决了文件的共享使用，还带来了隐藏文件路径、增加权限安全及节省存储等好处。若一个 inode 号对应多个文件名，则称这些文件为硬链接。换言之，硬链接就是同一个文件使用了多个别名（见 [图 2.](https://www.ibm.com/developerworks/cn/linux/l-cn-hardandsymb-links/#fig2)hard link 就是 file 的一个别名，他们有共同的 inode）。硬链接可由命令 link 或 ln 创建。如下是对文件 oldfile 创建硬链接。

由于硬链接是有着相同 inode 号仅文件名不同的文件，因此硬链接存在以下几点特性：

* 文件有相同的 inode 及 data block；
* 只能对已存在的文件进行创建；
* 不能交叉文件系统进行硬链接的创建；
* 不能对目录进行创建，只可对文件创建；
* 删除一个硬链接文件并不影响其他有相同 inode 号的文件。

# ls -li

total 0

// 只能对已存在的文件创建硬连接

# link old.file hard.link

link: cannot create link `hard.link' to `old.file': No such file or directory

# echo "This is an original file" > old.file

# cat old.file

This is an original file

# stat old.file

 File: `old.file'

 Size: 25           Blocks: 8          IO Block: 4096   regular file

Device: 807h/2055d      Inode: 660650      Links: 2

Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)

...

// 文件有相同的 inode 号以及 data block

# link old.file hard.link | ls -li

total 8

660650 -rw-r--r-- 2 root root 25 Sep  1 17:44 hard.link

660650 -rw-r--r-- 2 root root 25 Sep  1 17:44 old.file

// 不能交叉文件系统

# ln /dev/input/event5 /root/bfile.txt

ln: failed to create hard link `/root/bfile.txt' => `/dev/input/event5':

Invalid cross-device link

// 不能对目录进行创建硬连接

# mkdir -p old.dir/test

# ln old.dir/ hardlink.dir

ln: `old.dir/': hard link not allowed for directory

# ls -iF

660650 hard.link  657948 old.dir/  660650 old.file

清单 5. 查找有相同 inode 号的文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | # df -i --print-type  Filesystem     Type       Inodes  IUsed    IFree IUse% Mounted on  /dev/sda7      ext4      3147760 283483  2864277   10% /  udev           devtmpfs   496088    553   495535    1% /dev  tmpfs          tmpfs      499006    491   498515    1% /run  none           tmpfs      499006      3   499003    1% /run/lock  none           tmpfs      499006     15   498991    1% /run/shm  /dev/sda6      fuseblk  74383900   4786 74379114    1% /media/DiskE  /dev/sda8      fuseblk  29524592  19939 29504653    1% /media/DiskF    # find / -inum 1114  /media/DiskE/Pictures/t3.jpg  /media/DiskF/123.txt  /bin/sync |

软链接与硬链接不同，若文件用户数据块中存放的内容是另一文件的路径名的指向，则该文件就是软连接。软链接就是一个普通文件，只是数据块内容有点特殊。软链接有着自己的 inode 号以及用户数据块（见 [图 2.](https://www.ibm.com/developerworks/cn/linux/l-cn-hardandsymb-links/#fig2)）。因此软链接的创建与使用没有类似硬链接的诸多限制：

* 软链接有自己的文件属性及权限等；
* 可对不存在的文件或目录创建软链接；
* 软链接可交叉文件系统；
* 软链接可对文件或目录创建；
* 创建软链接时，链接计数 i\_nlink 不会增加；
* 删除软链接并不影响被指向的文件，但若被指向的原文件被删除，则相关软连接被称为死链接（即 dangling link，若被指向路径文件被重新创建，死链接可恢复为正常的软链接）。

# ls -li

 total 0

 // 可对不存在的文件创建软链接

 # ln -s old.file soft.link

 # ls -liF

 total 0

 789467 lrwxrwxrwx 1 root root 8 Sep  1 18:00 soft.link -> old.file

 // 由于被指向的文件不存在，此时的软链接 soft.link 就是死链接

 # cat soft.link

 cat: soft.link: No such file or directory

 // 创建被指向的文件 old.file，soft.link 恢复成正常的软链接

 # echo "This is an original file\_A" >> old.file

 # cat soft.link

 This is an original file\_A

 // 对不存在的目录创建软链接

 # ln -s old.dir soft.link.dir

 # mkdir -p old.dir/test

 # tree . -F --inodes

 .

├── [ 789497]  old.dir/

│   └── [ 789498]  test/

├── [ 789495]  old.file

├── [ 789495]  soft.link -> old.file

└── [ 789497]  soft.link.dir -> old.dir/

清单 8. 使用命令 find 查找软链接与硬链接

// 查找在路径 /home 下的文件 data.txt 的软链接

# find /home -lname data.txt

/home/harris/debug/test2/a

// 查看路径 /home 有相同 inode 的所有硬链接

# find /home -samefile /home/harris/debug/test3/old.file

/home/harris/debug/test3/hard.link

/home/harris/debug/test3/old.file

# find /home -inum 660650

/home/harris/debug/test3/hard.link

/home/harris/debug/test3/old.file

// 列出路径 /home/harris/debug/ 下的所有软链接文件

# find /home/harris/debug/ -type l -ls

656662 0 lrwxrwxrwx 1 harris harris 1 Sep 1 14:37 /home/harris/debug/test2/b -> a

656627 0 lrwxrwxrwx 1 harris harris 8 Sep 1 14:37 /home/harris/debug/test2/a ->

data.txt

789467 0 lrwxrwxrwx 1 root root 8 Sep 1 18:00 /home/harris/debug/test/soft.link ->

old.file

789496    0 lrwxrwxrwx   1 root     root            7 Sep  1 18:01

/home/harris/debug/test/soft.link.dir -> old.dir

（10） 文件权限怎么看（rwx）  
（11） 文件的三种时间（mtime, atime，ctime），分别在什么时候会改变  
（12） Linux监控网络带宽的命令，查看特定进程的占用网络资源情况命令