#关于命令执行：

用户所能使用的命令是依据PATH环境变量所规定的的目录去查找的。

| | |
|---|---|
| ; | 在前一个命令结束时，而忽略其返回值，继续执行下一个命令。 * |
| && | 在前一个命令结束时，若返回值为 true，继续执行下一个命令。 * |
| \|\| | 在前一个命令结束时，若返回值为 false，继续执行下一个命令。 * |

#语言

研究一门语言就是研究它如何把变量，字面量到表达式到语句到函数到组件再到包组合在一起的，组合的规则有那些，比如运算符，参数定义，组建定义及继承关系等等，以及该语言的并发是通过Thread提供还是语言原生支持，如yiled，async/await等。

#常出错误：

修正某个软件的错误需要创建一些新的文件，但因为软件启动的用户和修错时使用的不是同一个用户，经常导致新添加文件的访问权限对软件的启动用户权限不对。

在Linux或者Unix系统中，通过rm或者文件管理器删除文件将会从文件系统的目录结构上解除链接(unlink)；但是如果文件正在被一个进程使用，那么该进程将仍然可以读取该文件，磁盘空间也一直被占用。

#问题定位

TCP连接的初始化有方向性（inboud，outbound），需要注意测试的方向性。
502，503的问题Nginx不一定有日志输出，请使用curl -vvvv看status code。
http短连接通常通过 netstat -lnpt观察不到，必须使用curl -vvvv观察。
http排错请一定要使用curl -vvvv。

#80端口

Linux/OSX系统下需要root权限才能创建侦听小于1024端口的进程。

#性能数据

一般来说，千兆网卡上TCP的最大传输能力约为100MB/s，万兆网卡上TCP的最大传输能力约为1000MB/s。

#网络安全组规则5元组

Dir：inboud or outbound
源：Net or IP，Port
目标 Net or IP，Port

#流量监控

Glasswire

#DNS

阿里巴巴 (72.360, 0.770, 1.08%)宣布AliDNS公共解析服务正式上线，服务IP也比较好记，分别是223.5.5.5和223.6.6.6

#Private IP

**A: 10.0.0.0~10.255.255.255 即10.0.0.0/8**

**B:172.16.0.0~172.31.255.255即172.16.0.0/12**

**C:192.168.0.0~192.168.255.255 即192.168.0.0/16**

#阿里巴巴开源镜像站

https://opsx.alibaba.com

#nignx openresty

502 - Bad Gateway（坏的网关）,一般是网关服务器请求后端服务时，后端服务没有按照http协议正确返回结果，但在某些情况下，上游服务的超时（触发 tcp reset）也可能引发 502。 - the server returned an invalid or incomplete response => HTTP 502 - The server was acting as a gateway or proxy and received an invalid response from the upstream server

504 - Gateway Timeout（网关超时）,一般是网关服务器请求后端服务时，后端服务没有在特定的时间内完成服务。 - the server failed to reply in time => HTTP 504 - The server was acting as a gateway or proxy and did not receive a timely response from the upstream server

#关于控制键

TAB，命令和文件名补全；Ctrl + c 终止命令执行；Ctrl + d 退出控制台；Ctrl + r 搜索历史命令；Ctrl + z 后台执行；Ctrl + a 行首；Ctrl + e 行尾；Ctrl + k 删除到行尾；Ctrl + l 清除屏幕。

#ifconfig

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 192.168.12.111 netmask 255.255.240.0 broadcast 192.168.12.255 ether 00:16:3e:00:4e:31 txqueuelen 1000 (Ethernet) RX packets 3098106 bytes 1264308026 (1.1 GiB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 2513278 bytes 1230432102 (1.1 GiB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

第一行（网卡状态）：网卡名称：eth0，网卡状态：flags，mtu：最大传输单元第二行（三层信息）：IP地址：inet，掩码：netmask，广播：broadcast 第三行（二层信息）：MAC地址：ether，发送队列长度：txqueuelen 第四行（接收成功）：包：packets，字节：bytes 第五行（接收出错）：错误：errors，丢弃：dropped，过载：overruns，frame：帧数第六行（发送成功）：包：packets，字节：bytes 第七行（发送出错）：错误：errors，丢弃：dropped，过载：overruns，载波：carrier，碰撞：collisions

#java
jps -v 查看启动jvm时显示指定的参数
jinfo <pid> 查看VM flags && system props
java -XX:+PrintFlagsFinal 查看所有参数的默认值

#java线程数（有过3万）
线程最大数量由JVM的堆(-Xmx,-Xms)大小、Thread的栈(-Xss)内存大小、系统最大可创建的线程数的限制参数三个方面影响。不考虑系统限制，可以通过这个公式估算：线程数量 = (机器本身可用内存 - JVM分配的堆内存) / Xss的值。
-Xms 初始堆大小 (在实际生产中，一般把-Xms和-Xmx设置成一样的。)
-Xmx 最大堆大小
-Xss 每个线程栈大小

#sleep
    #bash
        sleep 60
    #nodejs
        function sleep(milliSeconds) {
            var startTime = new Date().getTime();
            while (new Date().getTime() < startTime + milliSeconds);
        };
        sleep(10000);
    #python
        import time
        //secs
        time.sleep(60)
    #go
        import time;
        time.Sleep(60s)
    #java

    #scala
#output log
    #nodejs
        console.log("")
    #python
        print("")
    #go

```
    #java
        System.out.println("")
    #scala
#debug
    #nodejs:
        #主动中断
        debugger
        #调试器模式启动
        node inspect script.js
    #python:
        #主动中断
        import pdb
        pdb.set_trace()#运行到这行程序会暂停
        #调试器模式启动
        python -m pdb script.py
    #go

    #java
        #主动中断
        无
        #调试器模式启动
        jdb
    #scala

#run sh
    #nodejs
    const exec = require('child_process').exec;
    var yourscript = exec('ls -al .', (error, stdout, stderr) => {
            console.log(stdout);
            console.log(stderr);
    });

    #python
        import subprocess;
        subprocess.call("env")
        subprocess.call("ls","-al",".")
    #go

    #java
```

#scala

#get stack info

#nodejs

#python

#go

#java

jstack <pid>

#scala

jstack <pid>

#profile

#nodejs

#python

#go

#java

#scalago

#数据库

查询：Index

模式：

主模式（聚集索引）

次要模式（非聚集索引【组合索引，覆盖索引, Where Index】）

统计：

：Index Size

：Index Usage

事务：

Begin Tran,

Commit

Rollback

explain:

usage：

explain***

plan cache：

：how to clear

: how to get

profile:

profiler***

#jdb

JDB, 意即Java Debuger，可以实现在命令行方式下的跟踪调试；JDWP（Java Debug Wire Protocol）Java调试网络协议，通过该协议Debuger（比如JDB）可以和目标VM通信，以便获取目标VM的包括类、对象、线程等信息。

#debug

#step 1 start target VM（suspend=y启动后程序会暂停到入口以便调试，suspend=n启动后便程序开始运行）

JAVA_OPTS="$JAVA_OPTS -Xdebug -Xrunjdwp:transport=dt_socket,address=5005,server=y,suspend=y"

java $JAVA_OPTS -jar abc.jar <params>

JAVA_OPTS="$JAVA_OPTS

-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005" java

$JAVA_OPTS -jar abc.jar <params>

#step 2 start jdb

jdb -attach 5005

#command:

threads

classes

class <class id>

locals

print <express>

print java.lang.System.getProperties()

stop at <class_full_name>:<line_number>

step

next

#Linux方法论

1.在使用一个命令的时候，最好要知道它属于哪个软件包；

2.会查找和安装软件包，安装软件包后，要能够找出文件都安装到哪里了；

3.要会阅读软件的文档；

4.实在不行了，那就看源代码吧。

#init

Linux 系统的内核加载完毕后，第一个执行的进程是 init，是进入字符界面还是进入图形界面，当然是 init 说了算。不同的系统，启动的第1个进程都叫 init，可此 init 等于彼 init 吗？所以，1.我们要找到这个 init 属于哪个软件包；2.找到这个软件包的文件都安装在文件系统的哪些位置，这个软件包还包含哪些程序，包含哪些文档；3.阅读这个软件包的文档（当然，首先利用搜索引擎对该软件包做一些功课是必须的）；4.实在不行就阅读源代码吧。

#init centos

在Red Hat流派的系统（CentOS、Fedora）中，软件的包管理器是 RPM。先看CentOS 5.10，首先，使用 which init 命令，可以查出 init的完整路径为 /sbin/init；然后使用 rpm -qf /sbin/init 命令，可以查出 init 程序所在的软件包为 SysVinit；最后，使用 rpm -ql SysVinit 命令，就可以看到这个软件包里面还有哪些程序、哪些文档以及它们分别位于文件系统的什么地方了

#init ubuntu

在 Ubuntu 中，首先，使用 which init 命令，可以查出 init 的完整路径为 /sbin/init；然后使用 sudo dpkg -S /sbin/init 命令，可以查出 init 程序所在的软件包为 upstart；最后，使用 sudo dpkg -L upstart 命令，就可以看到这个软件包里面还有哪些程序、哪些文档以及它们分别位于文件系统的什么地方了

Ubuntu系统：

- which
- dpkg -S
- dpkg -L
- aptitude search
- aptitude install
- apt-get source
- man
- info

CentOS、Fedora系统：

- which
- rpm -qf
- rpm -ql
- yum list
- yum install
- yumdownloader --source
- rpm2cpio
- cpio
- man
- info

#bash帮助

help

help if

help select

#切换用户不改变环境变量

su <user>

#切换用户并改变环境变量: -, -l , --login make the shell a login shell

su - <user>

su -l <use>

su --login <user>

#su和su - 命令使用最主要的区别主要是涉及login-shell和non-login shell的变量读取方法，两种不同的变量读取方法导致执行用户切换后HOME/PATH/USER/MAIL等变量不同。

#su命令执行后的pstree

在命令行执行 su - davis，然后pstree后部分结果

sshd(4939) -----sshd(73751)----bash(73770)-----su(74598)-----bash(74599)---pstree(74653)

#Ad Hoc Shell脚本的执行

bash -c "command string"

#Shell脚本文件的执行

Shell脚本文件通常是一个以shebang(#!)起始的文本文件。shebang是一个文本行，其中#!位于解释器路径之前。

有两种运行脚本文件的方式；一种是将脚本文件作为bash的命令行参数，另一种是授予脚本文件执行权限，将其变为可执行文件。

方式一：（脚本文件不需要有shebang）

bash /path/to/script.sh

方式二：（脚本文件需要有shebang，脚本要有执行权限）

/path/to/script.sh

#bash -l/--login

Make bash act as if it had been invoked as a login shell(A Login shell is one whose first character of argument zero is a -, or one started with the --login option)

在crontab中，定时执行自定义脚本，不加-l，则不会自动导入任何环境变量，为了导入JAVA_HOME环境变量，加-l，以便自动导入环境变量，导入文件的顺序如下：

/etc/profile（全局配置）

~/.bash_profile(通常配置去读~/.bashrc)

if前文件不存在 ---> ~/.bash_login

if 两者都不存在---> ~/.profile

suse中也读~/.bashrc

#测试上传/下载速度-使用scp

使用scp

#测试上传/下载速度-使用iperf

#服务器端： -i report interval, -l buffer size

iperf -s -p <port> -i 1 -l 128k

#客户端: -i report interval, -l buffer size, -t test time(second), -P test thread number

iperf -c <ip> -p <port> -i 1 -l 128k -P 2 -t 30

Linux中的5个用于审计的命令：

last, lastb - show listing of last logged in users

last：此命令查看系统中成功的登录，关机，重启等情况，此命令将/var/log/wtmp文件格式化輸出。

lastb：此命令查看登录失败的情况，此命令将/var/log/btmp文件格式化輸出。

lastlog：此命令用于查看用户上一次的登录情况，此命令将/var/log/lastlog文件格式化輸出。

who：此命令查看当前登录系统的情况，此命令将/var/log/utmp文件格式化輸出。

w：与who命令一致。

#sudo - execute a command as another user

sudo <command args> #run command as root

sudo -u <user> <command args> #run command as <user>

•sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The default security policy is sudoers, which is configured via the file /etc/sudoers, or via LDAP.

•sudo可让用户以其他的身份来执行指定的指令，预设的身份为root。在/etc/sudoers中设置了可执行sudo指令的用户。若其未经授权的用户企图使用sudo，则会发出警告的邮件给管理员。用户使用sudo时，必须先输入密码，之后有5分钟的有效期限，超过期限则必须重新输入密码。

-i，--login 以目标用户（未使用-u指定则为root）身份运行一个 登陆Shell

-s，--shell 以目标用户（未使用-u指定则为root）身份运行一个 Shell

#runlevel -- find the previous and current system runlevel.

sudo runlevel

#rcconf - Debian Runlevel configuration tool

#list services

rcconf —list

#on service

rcconf —on nginx

#off service

rcconf —off nginx

系统负载 / CPU 负载 – 衡量 Linux 系统的 CPU 过载或利用率低的指标，即处于运算状态或等待状态的 CPU 核心数。

平均负载 – 通过固定的时间周期如 1、5、15 分钟计算出的平均的系统负载。

假设系统负载如下：

23:16:49 up 10:49, 5 user, load average: 1.00, 0.40, 3.35

在单核系统中意味着：

CPU 被充分利用（100%）；最近的 1 分钟有 1 个进程在运行。

CPU 有 60% 处于空闲状态；在最近的 5 分钟没有进程等待 CPU 时间。

CPU 平均过载了 235%；最近的 15 分钟平均有 2.35 个进程在等待 CPU 时间。

在双核系统中意味着：

有一个 CPU 处于完全空闲状态，另一个 CPU 被使用；最近的 1 分钟没有进程等待 CPU 时间。

CPU 平均 160% 处于空闲状态；最近的 5 分钟没有进程等待 CPU 时间。

CPU 平均过载了 135%；最近的 15 分钟有 1.35 个进程等待 CPU 时间。

#watch - execute a program periodically, showing output fullscreen

watch [options] command

watch -n 1 "date"

#timedatectl

timedatectl

#all timezone

/usr/share/zoneinfo/*

#current timezone

cat /etc/timezone

#current timezone by date

date -R

#current timezone by link

ll /etc/localtime

#set timezone by link

sudo unlink /etc/localtime

sudo ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

#set timezone by tzselect

tzselect

#无限循环

while [[ 0 ]]; do sleep 1; ps -ef|grep phantomjs; done

while [[ 0 ]]; do date "+%y-%m-%d %H:%M:%S.%n"; done

while [[ 0 ]]; do openssl rand -base64 35; done

#linux与windows文件格式转换

dos2u (0.390, -0.017, -4.18%)nix

unix2dos

#uname - print system information

#查看linux内核版本:

1. cat /proc/version

2. uname -a

3. uname -r

#查看linux版本:

1. lsb_release -a

2. cat /etc/issue

3. cat /etc/os-release

#namespace

#查看当前进程的namespace

lsns

#查看某个进程的namespace

sudo lsns -p <pid>

#colums

NS namespace identifier (inode number)

TYPE kind of namespace

PATH path to the namespace

NPROCS number of processes in the namespace

PID lowest PID in the namespace

PPID PPID of the PID

COMMAND command line of the PID

UID UID of the PID

USER username of the PID

#net namesapce相关

ip netns add <namespace name> # 添加network namespace

ip netns list # 查看Network Namespace

ip netns delete <namespace name> # 删除Network Namespace

ip netns exec <namespace name> <command> # 进入到Network Namespace当中执行命令

#cgroup v1

#for total
/proc/cgroups/*/*
#for pid
cat /proc/<pid>/cgroup
#cgroup v2:
/sys/fs/cgroup/cgroup.controllers

#查看cpu信息
npoc
lscpu
arch

#查看当前shell的pid
echo $$

#dpkg(yum)
dpkg -i <pack>

#dpkg - package manager for Debian
dpkg -i <filename.deb>
dpkg-query actions
-l, --list package-name-pattern…
List packages matching given pattern.
dpkg -l
-s, --status package-name…
Report status of specified package.
dpkg -s lms

#rpm(apt)
rpm -ivh  samba-3.0.10-1.4E.i386.rpm //按路径安装并显示进度

#具体安装了那些包
rpm -qa

#卸载软件包
rpm -e <package>

#inotify
apt-get --yes install inotify-tools
apt-get --yes install  libnotify-bin

```
inotifywait -e modify $HOME/foo

#run this specific application as this specific user and get out of the pipeline
gosu

#列出所有块设备
lsblk

#服务开机自启动配置（CentOS）
chkconfig --list
chkconfig --list mysql
chkconfig mysql on
chkconfig mysql off

#checksum
md5sum <file>
sha1sum <file>  && shasum <file>
sha224sum <file> && shasum -a 224 <file>
sha256sum <file> && shasum -a 256 <file>
sha384sum <file> && shasum -a 384 <file>
sha512sum <file> && shasum -a 512 <file>pd

#pmap - report memory map of a process
pmap <pid>

#stat
stat <file>
#Count all open files
lsof|wc -l
#The tee utility copies standard input to standard output, making a copy in zero or more files. The output is unbuffered. 代替>重定向（有些平台限制>的使用）
tee [-ai] [file ...]
echo "$(date) [INFO] start to install app" | tee -a ${bin_folder}/logs/start.log

#BusyBox (28.320, -0.080, -0.28%) 是一个集成了三百多个最常用Linux命令和工具的软件.
bozybox [command] [args]

#对付病毒
nohup watch -n5 "crontab -r; chmod -R -x /tmp; busybox ps -ef|busybox grep /tmp/|busybox cut -c 0-6|xargs busybox kill -9"&
```

#pgrep,  pkill - look up or signal processes based on name and other attributes

pgrep [-Lafilnoqvx] [-F pidfile] [-G gid] [-P ppid] [-U uid] [-d delim] [-g pgrp]

[-t tty] [-u euid] pattern …

-l Long output.

-i Ignore case distinctions in both the process table and the supplied pattern.

-f Match against full argument lists. The default is to match against process names.

-x Require an exact match of the process name, or argument list if -f is given.

pkill [-signal] [-ILafilnovx] [-F pidfile] [-G gid] [-P ppid] [-U uid] [-g pgrp]

[-t tty] [-u euid] pattern …

#search file in folder(only show filename, ignore case)

grep -lir /folder/to/search/*

#match against process names, -l show process name

pgrep -lx ssh

pkill -x ssh

pgrep -lfx "java -jar build/libs/learning-game-api-0.0.1.jar"

pkill -fx "java -jar build/libs/learning-game-api-0.0.1.jar

#Parallel Distributed Shell-pdsh

pdsh

#split file

split [option] [input [prefix]]

split -b 100m file abc

#merge file

cat prefix* > marged_file

#match against full argument lists, -l will output process name and full arguments

pgrep -lf ssh

pgrep -lf ’ssh -N -D 1080 root@45.32.17.22'

pkill -lf  ’ssh -N -D 1080 root@45.32.17.22'

#tr -- translate characters

tr [-Ccsu] string1 string2

tr [-Ccu] -d string1

tr [-Ccu] -s string1

tr [-Ccu] -ds string1 string2

#ipcmk - make various IPC resources

ipcmk

#ipcs - show information on IPC facilities

ipcs

#iprm

ipcrm

#history && !

history

history|grep “scp”

!! #最后一跳命令

!211 #history中第211号命令

!-2 #倒数第2条命令

!^ #最后一条命令的第一个参数

![cmd]:[no#] #最后一条命令的第no个参数

!$ #最后一条命令的最后一个参数

!* #最后一条命令的所有参数

! [keywords] #根据keywords查找执行历史命令

#以sudo执行最后一条命令

sudo !!

#查看本地的字符集

locale -a

#查看所有支持的字符集

locale -m

#查看当前默认设置

echo $LANG

#字符集的设置:

1. 临时 LANG="zh_CN.UTF-8" export LANG= "zh_CN.UTF-8"

2. 修改/etc/sysconf/i18n 或 /etc/locale.conf文件，把 LANG="zh_CN.UTF-8"

#change shell

chsh[-s<shell名称>][用户名称]

#系统级配置

#sysctl - configure kernel parameters at runtime

sysctl -a

*#cat /proc/sys/fs/file-max*

#/etc/sysctl.conf ——> sysctl -p

#sysctl与内核参数及/proc/sys/*

#打开Linux路由功能

sysctl net.ipv4.ip_forward=1

#关闭Linux路由功能

sysctl net.ipv4.ip_forward=0

#sshd

man sshd_config

vim /etc/ssh/sshd_config

#GatewayPorts yes

service ssh reload

#pssh

在多台机器上批量执行命令的工具。

pssh ...

pscp ...

#用户级

#ulimit - get and set user limits

ulimit有软限制和硬限制之分，硬限制用-H，软限制用-S参数

#get

ulimit -a #show all limit for soft

ulimit -aH #show all limit for hard

ulimit -n #show max open files for soft

ulimit -nH #show max open files for hard

#set

ulimit -n 1024 #set max open files both soft and hard

ulimit -nS 1024 #set max open files only soft

ulimit -nH 1024 #set max open files only hard

#lsof - list open files(lsof)

lsof | lsof -p pid

#select by IPv[46] address

lsof -i:<port>

#list all listen ports

lsof -i -P

#select fifo

lsof | grep FIFO

#获得一个已经被删除但是仍然被应用程序占用的文件列表
lsof |grep deleted
#获取监听列表
#lsof -i -P
#lsof |GREP LISTEN
#lsof -P |GREP LISTEN

#trace system calls and signals(apt-get install strace)
strace常用来跟踪进程执行时的系统调用和所接收的信号，strace在底层使用内核的ptrace特性来实现其功能。 在Linux世界，进程不能直接访问硬件设备，当进程需要访问硬件设备(比如读取磁盘文件，接收网络数据等等)时，必须由用户态模式切换至内核态模式，通过系统调用访问硬件设备。strace可以跟踪到一个进程产生的系统调用,包括参数，返回值，执行消耗的时间。
在计算机中，系统调用指运行在用户空间的程序向操作系统内核请求需要更高权限运行的服务。系统调用提供用户程序与操作系统之间的接口。操作系统的进程空间分为用户空间和内核空间：操作系统内核直接运行在硬件上，提供设备管理、内存管理、任务调度等功能。用户空间通过API请求内核空间的服务来完成其功能——内核提供给用户空间的这些API, 就是系统调用。

#系统调用相关
在Linux系统上，应用代码通过glibc库封装的函数，间接使用系统调用。Linux内核目前有300多个系统调用，详细的列表可以通过syscalls手册页查看。这些系统调用主要分为几类：
文件和设备访问类 比如open/close/read/write/chmod等
进程管理类 fork/clone/execve/exit/getpid等
信号类 signal/sigaction/kill 等
内存管理 brk/mmap/mlock等
进程间通信IPC shmget/semget * 信号量，共享内存，消息队列等
网络通信 socket/connect/sendto/sendmsg 等
其他

man syscalls
man dup2

#shell history
export HISTSIZE="100000"

strace有两种运行模式：一种是通过它启动要跟踪的进程，用法很简单，在原本的命令前加上strace即可。另外一种运行模式，是跟踪已经在运行的进程，在不中断进程执行的情况下，理解它在干嘛。 这种情况，给strace传递个-p pid 选项即可。

-c 统计每一系统调用的所执行的时间,次数和出错的次数等.
-d 输出strace关于标准错误的调试信息.

-f 跟踪由fork调用所产生的子进程.

-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号.

-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪.

-h 输出简要的帮助信息.

-i 输出系统调用的入口指针.

-q 禁止输出关于脱离的消息.

-r 打印出相对时间关于,,每一个系统调用.

-t 在输出中的每一行前加上时间信息.

-tt 在输出中的每一行前加上时间信息,微秒级.

-ttt 微秒级输出,以秒了表示时间.

-T 显示每一调用所耗的时间.

-v 输出所有的系统调用.一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出.

-V 输出strace的版本信息.

-x 以十六进制形式输出非标准字符串

-xx 所有字符串以十六进制形式输出.

-o filename 将strace的输出写入文件filename

-p pid 跟踪指定的进程pid.

-s strsize 指定输出的字符串的最大长度.默认为32.文件名一直全部输出.

-u username 以username 的UID和GID执行被跟踪的命令

实例：跟踪nginx, 看其启动时都访问了哪些文件

strace -tt -T -f -e trace=file -o /data/log/strace.log -s 1024 ./nginx

实例：定位进程异常退出

问题：机器上有个叫做run.sh的常驻脚本，运行一分钟后会死掉。需要查出死因。

定位：进程还在运行时，通过ps命令获取其pid, 假设我们得到的pid是24298

strace -o strace.log -tt -p 24298

查看strace.log, 我们在最后2行看到如下内容:

22:47:42.803937 wait4(-1, <unfinished ...>

22:47:43.228422 +++ killed by SIGKILL +++

strace -v -s 4096 -p <pid>

strace -v -s 4096 <program> [args]

#计算系统调用的时间并打印出来

strace -cfp $(pidof mysqld)

#mkfifo -- make fifos

mkfifo abc

#at shell 1

find * / > abc

#at shell 2

grep test < abc

#dnstracer

#latrace - traces library calls in dynamically linked programs

#leaktracer - Simple and efficient memory-leak tracer for C++ programs

#ltrace - Tracks runtime library calls in dynamically linked programs

#pstack - Display stack trace of a running process

#pythontracer - Python programs' execution tracer and profiler

#pytimechart - GUI View (1.150, 0.000, 0.00%)er for Linux kernel traces

#python-tracer - Centralized trace management using sys.settrace

#tcptrace - Tool for analyzing tcpdump output

#tcptraceroute - traceroute implementation using TCP packets

#traceroute - Traces the route taken by packets over an IPv4/IPv6 network

#sar - Collect, report, or save system activity information.

sar -n DEV 1

#alias 查看当前shell中定义的所有alias或者在当前shell中定义一个alias

usage: alias [-p] [name[=value] ... ]

#sed - stream editor for filtering and transforming text - (两种调用形式)

sed [-Ealn] command [file ...]

sed [-Ealn] [-e command] [-f command_file] [-i extension] [file ...]

The form of a sed command is as follows:

[address[,address]]function[arguments]

Whitespace may be inserted before the first address and the function portions of the command.

删除：d命令

sed '$d' example-----删除example文件的最后一行。

sed '/test/'d example-----删除example文件所有包含test的行。

sed '\:/usr/local/openresty/bin/openresty:d' rc.local --删除rc.local中包含 /usr/local/openresty/bin/openresty的行

替换：s命令

sed 's/test/mytest/g' example-----在整行范围内把test替换为mytest。如果没有g标记，则只有每行第一个匹配的test被替换成mytest。

sed -i 's@test@mytest@g' example-----在整行范围内把test替换为mytest。如果没有g标记，则只有每行第一个匹配的test被替换成mytest

多点编辑：e命令

写入文件：w命令

sed '/test/r file' example-----file里的内容被读进来，显示在与test匹配的行后面，如果匹配多行，则file的内容将显示在所有匹配行的下面。

追加命令：a命令

插入：i命令

从文件读入：r命令

下一个：n命令

sed '/test/{ n; s/aa/bb/; }' example-----如果test被匹配，则移动到匹配行的下一行，替换这一行的aa，变为bb，并打印该行，然后继续。

变形：y命令

退出：q命令

sed '10q' example-----打印完第10行后，退出sed。

## # sed -n '/^起始行/,/^结束行/p' FILE_NAME

#awk程序的工作流程是：逐行扫描文件，从第一行到最后一行，寻找匹配特定模式的行，并在这些行上进行用户想要的操作。awk的基本结构由模式匹配和处理动作组成，awk读取文件内容的每一行时，将比对该行是否与给定的模式匹配，如果匹配则执行处理过程，否则对该行不做任何处理。如果没有指定处理动作，默认动作为print打印行；如果没有指定模式匹配，则默认匹配所有数据。

#awk程序的语法结构是：一个awk程序包含一系列的模式{动作指令}或是函数定义，模式可以是BEGIN、END、表达式，用来限定操作对象的多个表示式使用逗号分隔；动作指令需要以{}引起来。

#awk有两个特殊的模式匹配：BEGIN和END，它们被放置在没有读取任何数据之前以及在所有数据读取完成以后执行。

awk '/^root/ {print $1} /^nobody/ {print $1}' /etc/passwd

ps -ef|grep "sleep 99999"|grep -v 'grep'|awk '{print $2}'|xargs -r kill -9

ps -ef|grep "sleep 99999"|grep -v 'grep'|awk '{system("kill -9 "$2)}'

awk "NR==2"取第二行

awk "NR>0"去掉第一行

# check command line

cat /proc/<pid>/cmdline|tr "\0" "\n"

#清除openresy进程使用ps -e列出进程的执行文件名，这样不会清除其他nginx进程

ps -e|grep "openresty"|grep -v 'grep'|awk '{print $1}'|xargs -r kill -9

#pstree - display a tree of processes - 默认不使用-p参数会显示进程的线程数

pstree [options]

pstree [options] <pid>

pstree [options] <user>

options:

-p show pid以及线程id

-u 显示uid变化

-a 显示参数

#install pstree

apt-get install psmisc

#xargs - build and execute command lines from standard input

which vim|xargs ls -al

#docker image ls|grep none|cut -c 77-100| xargs docker rmi

#nslookup - query Internet name servers interactively

nslookup www.163.com

#cut只擅长处理“以一个字符间隔”或者固定长度的，如果多个不等空格分割就不行了

#cut 按域(:)

cat /etc/passwd|cut -d : -f 1

#cut 按域(制表符)

cut -f 1 a.txt

#固定长度

who | cut -c 1-16,26-38

who | cut -c 26-38

#查看文件中是制表符还是空格

sed -n l a.txt

#通过sort/uniq去重查看access日志，以便了解那个时间段服务中断服务

cat **/interfaceLog/****.2020-12-30_1.log|cut -c 1-22|grep "2020-12-30 16:"|sort|uniq -c

#http

httpie - CLI, cURL-like tool for humans

SYNOPSIS

http [-h] [--version] [--json | --form] [--traceback]
[--pretty ] [--headers | --body] [--style STYLE]
[--auth AUTH] [--verify VERIFY] [--proxy PROXY]
[--file PATH] [--timeout TIM (17.750, -0.280, -1.55%)EOUT]
METHOD URL [items [items ...]]

Positional arguments:

METHOD HTTP method to be used for the request (GET, POST, PUT, DELETE, PATCH, ...).

URL Protocol defaults to http:// if the URL does not include it.

items HTTP header (key:value), data field (key=value) or raw JSON field (field:=value).

#word count(byte,line,word)
wc [-clmw] [file...]

#sha1sum - compute and check SHA1 message digest
sha1sum [OPTION]... [FILE]...

#全面了解系统
df , ps , free, uptime, last reboot

#free - Display amount of free and used memory in the system
free -h

#df - report file system disk space usage
df -h

#uptime - Tell how long the system has been running.
uptime

#time(Shell中的，不支持参数)
time cmd [args...]

#/usr/bin/time
/usr/bin/time [options] cmd [args...]

#du - estimate file space usage
du -hc .

du  - hcd1 .

du -h --max-depth=1 .

du -hs .

#gzip

gzip <file>

gzip -d <file>

#ps - report a snapshot of the current processes.(procps)

ps

ps -A|e

ps au|aux|ax

#显示所有进程

ps -efj

#显示指定程序的进程

ps -f -C java -o pid=

#显示指定进程的线程(thread)

ps -fL -p <pid>

ps --forest

#taskset - retrieve or set a process's CPU affinity

#retrieve

taskset -p <pid>

#set

taskset -p <mask> <pid>

#显示最近的10条系统消息

#print or control the kernel ring buffer

dmesg | tail

#Report virtual memory statistics

vmstat

vmstat 1#(每1秒)

#iostat - Report Central Processing Unit (CPU) statistics and input/output statistics for devices and partitions.

iostat

iostat 1#(每1秒)

#网络统计

vnstat -l

#perf - performance analysis tools for Linux
perf [--version] [--help] [OPTIONS] COMMAND [ARGS]

#perf-top - System profiling tool.
#This command generates and displays a performance counter profile in real time.
perf top [-e <EVENT> | --event=EVENT] [<options>]

#perf-stat - Run a command and gather performance counter statistics
perf stat [-e <EVENT> | --event=EVENT] [-a] <command>
perf stat [-e <EVENT> | --event=EVENT] [-a] — <command> [<options>]
perf stat [-e <EVENT> | --event=EVENT] [-a] record [-o file] — <command> [<options>]
perf stat report [-i file]

#sysstat 软件包
apt-get install sysstat

#pidstat - Report statistics for Linux tasks.
pidstat 1#(每1秒)

#mpstat - Report processors related statistics.
mpstat

#supervisor - 后台进程管理工具
supervisord：supervisor的服务器端部分
supervisorctl：启动supervisor的命令行窗口
config:
    /etc/supervisor/supervisord.conf
    /etc/supervisor/conf.d/programxxx.conf

    [program:lms]
    directory = /opt/apps/lms
    command = /opt/apps/lms/virtualenv/bin/newrelic-admin run-program /opt/apps/lms/virtualenv
    /bin/python run_with_tornado.py -p 8000 -n 8 -c any
    autostart = false
    user = lms
    stopsignal = QUIT
    stdout_logfile = /var/log/lms/lms-stdout.log

```
stderr_logfile = /var/log/lms/lms-stderr.log
environment = CONF="/opt/conf/lms/prod.py",NEW_RELIC_CONFIG_FILE="/opt/conf
/lms/newrelic.ini"

supervisorctl status

supervisorctl
stop
programxxx
supervisorctl start/stop/restart programxxx
supervisorctl stop all
supervisorctl reload
supervisorctl reread

#env - run a program in a modified environment
#Set each NAME to VALUE in the environment and run COMMAND.
env [OPTION]... [-] [NAME=VALUE]... [COMMAND [ARG]...]
#If no COMMAND, print the resulting environment.
env

#get env of process
cat /proc/<pid>/environ | tr "\0" "\n"

#lsmod - Show the status of modules in the Linux Kernel
lsmod

#使用openssl对文件生成sha256 摘要签名，并进行 Base64 编码
curl https://example.com/static/js/other/zepto.js | openssl dgst -sha256 -binary | openssl enc -base64 -A

#从二机制文件创建base64
openssl enc -base64 -A -in abx.xlsx -out abc_base64.txt

#从base64生成二进制文件
openssl enc -base64 -A -d-in abc_base64.txt -out abc.xlsx

#以hex查看二进制文件
hexdump <binfile>
xxd <binfile>
```

证书相关的文件扩展名有CRT/CER/KEY/CSR/P12/JKS/PEM/DER，有的代表证书，有的代表私

钥，有的可以合二为一，并且扩展名和编码方式不是一一对应的。

| 证书类型 | 证书格式 | 证书扩展名 | 编码方式 |
| --- | --- | --- | --- |
| 设备数字证书 | PEM | .pem .cer .crt .der | ASCII(Base64) |
| | DER | .der .cer .crt | 二进制编码 |
| | PFX（带私钥） | .pfx .p12 | |
| CA证书 | PEM | .pem .cer .crt .der | ASCII(Base64) |
| | DER | .der .cer .crt | 二进制编码 |
| PKCS#12（P12） | 证书库，后缀名可以是.p12或.pfx。一般来说可以包含一个证书和一个私钥，也可以包含多个证书。 | | |
| keystore | Java独有的证书库，后缀名分为JKS/JCEKS/PKCS12（p12）等格式。 | | |

#私钥：

#证书

标准：

X.509

编码格式：

PEM - Privacy Enhanced Mail，文本格式，以-----BEGIN...开头，以-----END...结尾，内容为Base64字符串

使用：Apache和Linux服务器偏向于使用这种格式。

查看：openssl x509 -in <file>.pem -text -noout

DER - Distinguished Encoding Rules，二进制格式

使用：Java和Windows服务器偏向于使用这种格式。

查看：openssl x509 -in <file>.der -inform der -text -noout

相关文件：

*.crt 证书文件，常见于Linux系统，有可能是PEM编码，也可能是DER编码的，大多数应该是PEM编码的。

*.cer 证书文件，常见于Windows系统，有可能是PEM编码，也可能是DER编码的，大多数应该是DER编码的。

*.csr Certificate Signing Re

# 查看KEY信息

> openssl rsa -noout -text -in myserver.key

```
# 查看CSR信息
> openssl req -noout -text -in myserver.csr

# 查看证书信息
> openssl x509 -noout -text -in ca.crt

# 验证证书
# 会提示self signed
> openssl verify selfsign.crt

# 因为myserver.crt 是幅ca.crt发布的，所以会验证成功
> openssl verify -CAfile ca.crt myserver.crt

#openssl中的ciphers
openssl ciphers -V

#jvm中的ciphers(jdk 14)
keytool -showinfo -tls

#从https URL下载pem证书
openssl s_client -showcerts -connect www.163.com:443 2>&1 </dev/null | sed -n
'/^-----BEGIN CERTIFICATE-----/,/^-----END CERTIFICATE-----/p' > server_crt.pem

keytool -printcert -sslserver www.163.com:443 -rfc > server_crt.pem

#安装Java证书（从pem）
openssl x509 -in server_crt.pem -outform der -out server.crt
keytool -import - trustcacerts -keystore /opt/**/jdk/jre/lib/security/cacerts -storepass
123456
-noprompt -alias server -file server.crt

#查看openssl提供的加密套件suite
openssl ciphers -v

#查看keystore内容
keytool -list -v -keystore <file> -storepass changeit -storetype PKCS12

#nc
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound: nc -l -p port [-options] [hostname] [port]
```

@receiver

nc -l -p PORT > target_file

@sender

nc RECEIVER_HOST PORT < source_file

#start http server with python2 in current folder

python -m SimpleHTTPServer [port]

<port:8000>

#start http server with python3 in current folder

python3 -m http.server [port]

<port:8000> [—bind <ip>]

#查看系统的所有环境变量和自定义变量

#Set or unset values of shell options and positional parameters.

set

set -e

set -o pipefail

help set

#PWD与CWD

pwd是一条Shell命令，用来打印此Shell进程的CWD；

cwd是进程的当前工作目录，可以使用lsof -p <pid>查看，也可以ls -al /proc/<pid>/cwd 查看软连接。

#设置环境变量(环境变量一般全大写)

export ABC=123465

export PATH=/usr/share/gradle/bin:$PATH

#设置自定义变量

dee=123458

#显示信息或变量(环境或自定义)值

echo hello

echo $abc

echo $PATH

#hostname

hostname

#ip

hostname -i

#man capabilities

capget, capset - set/get capabilities of thread(s)

#type命令用来显示指定命令的类型，一个命令的类型可以是如下之一：alias （别名）、keyword（ 关键字，Shell保留字）、function（函数，Shell函数）、builtin（内建命令，Shell内建命令）、file （文件，磁盘文件，外部命令）、unfound（没有找到）。它是Linux系统的一种自省机制，知道了是那种类型，我们就可以针对性的获取帮助。比如内建命令可以用help命令来获取帮助，外部命令用man或者info来获取帮助。

type <cmd>  ：命令的基本使用方式就是直接跟上命令名字。

type -a <cmd> ：可以显示所有可能的类型，比如有些命令如pwd是shell内建命令，也可以是外部命令。

type -p <cmd> ：只返回外部命令的信息，相当于which命令。

type -f <cmd> ：只返回shell函数的信息。

type -t <cmd> ：只返回指定类型的信息。

#file命令用来查看文件的类型信息，比如可执行文件，脚本，动态连接库等

file <file>

#ldd命令用来显示可执行文件用到的动态连接库信息

ldd <elf_file>

#objdump

objdump -t /lib64/libc.so|grep dup2

#nm

nm -g /lib64/libc.so|grep dup2

#binutils

rpm -ql binutils

#常驻运行程序方式一

使用sysV或systemd的方式

#常驻运行程序方式二

tmux, nohup, screen

#常驻运行程序方式三使用deamon tools

Python的supervisord，Perl的ubic，Ruby的god。

#nohup - run a command immune to hangups, with output to a non-tty

#1. nohup with default log file nohup.log

nohup <command> &

nohup <command> <args-of-command> &

#2. nohup with specified log file

nohup command > mylog.txt 2>&1 &

nohup command args-of-command > mylog.txt 2>&1 &

#screen

1. create screen

screen -dmS myScreen

2. connect screen ( Ctl + A + D 断开环境 )

screen -r myScreen

3 list

screen list

#查找文件中包含foo的文件

ack foo

#同步文件夹

rsyrc

#输出重定向

> ; >>

ls > o.txt 2>&1

#输入重定向

< ; <<

#管道

|

#分页显示内容 ( 分页器 )

more , less , | more , | less

#管道命令

more , less , cut ( 分列 ) , grep ( 行过滤 )

netstat -lpnt | less

netstat -lpnt | grep "9000"

tail -n300 application.log |grep "Exception"

#,重启,登陆,退出

login , logout , shutdown , reboot , halt , poweroff

#退出Shell

exit

#显示登陆日志

last

#查看命令帮助文档

help set

man ls; q 退出

info ls; q 退出

#查看文件属性

ls ；尤其-d，-a，-l等参数特别重要

ls -al play*　　ls -al play-?

#查看目录树

tree [<dir>]

tree -L 3 [<dir>]

#命令查找

which javac

whereis javac

#查文件或目录

find /usr -name play

find /usr -name application.confne

find /usr -name play-2*

find /usr -name *play-2

find /usr -name *play-2*

find /usr -name "**play-2**”

find /usr -type f -name *.py -mtime -1

#find && delete

想find+rm可以这样 find . -name '*.py' -delete

find . -name .git -exec rm -fr {} \;

find . -name ‘*.py’ |xargs rm -rf

#delete all .git in any sub folder

rm -fvr ./**/.git

find . -name .git -exec echo {} \;

find . -name .git -exec rm -fr {} \;

#grep, egrep, fgrep, rgrep - print lines matching a pattern

grep [OPTIONS] PATTERN [FILE...]

grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

#文本内容查找

grep [OPTION]... PATTERN [FILE]...

-i, --ignore-case  Perform case insensitive matching.

-r, --recursive like --directories=recurse

grep -i 'hello world' menu.h main.c

grep -ir 'hello world' *

#设置当前用户的密码

passwd

#设置tom的用户密码

passwd tom

#设置tom的用户密码为abc123456

echo “abc123456” | passwd tom

echo -e “abc123456\nabc123456” | passwd tom

#锁定账户tom

passwd -l tom

#解锁账户tom

passwd -u tom

#清空账户tom密码（无密码可登陆系统）

passwd -d tom

#查看用户密码过期情况

chage -l <user>

#延期账号命令

chage -M 99999 <user>

#延期密码命令(-1，负一表示永不过期)

chage -E -1 <user>

#杀进程

kill -9 <pid>;

kill -KILL <pid>;

kill -15 <pid>;

kill -TERM <pid>;

kill -QUIT <pid>

killall

#打印调用栈

kill -3 <pid>;

#查看端口占用：

ss -lnpt

netstat -lnpt

lsof -i-P

lsof -i:8080

fuser -v 8080/tcp

nmap -sV -p 8080 localhost

#网络基础

网络传输中的三张表，MAC地址表（二层：交换机），ARP缓存表（三层到二层：路由器，PC），路由表（三层：路由器，PC）

路由表用于三层在发送IP数据包前，查找下一跳的IP地址，下一跳IP地址可能是Gateway（跨网段）也可能直接是目标IP（同一个网段内）。路由表有静态路由（手动设置）和动态路由（根据路由发现协议学习而得）。

ARP表用来查找下一跳IP（三层）对应的MAC地址（二层）的，ARP表由ARP协议主动获得。

MAC地址表用来查询MAC地址对应的交换机端口以便于交换机完成源端口到目标端口的数据交换，MAC地址表由交换机从端口接收到的二层数据帧的源MAC地址学习而得。

#查看路由表

route -n

Destination Gateway Genmask Flags Metric(未用) Ref(未用) Use(未用) Iface

| | |
|---|---|
| Destination | 目标网段或者主机 |
| Gateway | 网关地址，”*” 表示目标是本主机所 |
| Genmask | 网络掩码 |
| Flags | 标记。一些可能的标记如下： |
| | U — 路由是活动的 |
| | H — 目标是一个主机 |
| | G — 路由指向网关 |
| | R — 恢复动态路由产生的表项 |
| | D — 由路由的后台程序动态地安装 |

| | |
|---|---|
| | M — 由路由的后台程序修改 |
| | ! — 拒绝路由 |
| Metric | 路由距离，到达指定网络所需的中转 |
| Ref | 路由项引用次数（linux 内核中没有 |
| Use | 此路由项被路由软件查找的次数 |
| Iface | 该路由表项对应的输出接口 |

3 种路由类型

主机路由

主机路由是路由选择表中指向单个IP地址或主机名的路由记录。主机路由的Flags字段为H。例如，在下面的示例中，本地主机通过IP地址192.168.1.1的路由器到达IP地址为10.0.0.10的主机。

Destination Gateway Genmask Flags Metric Ref Use Iface

---------- ------- ------- ----- ------ --- --- -----

10.0.0.10 192.168.1.1 255.255.255.255 UH 0 0 0 eth0

网络路由

网络路由是代表主机可以到达的网络。网络路由的Flags字段为N。例如，在下面的示例中，本地主机将发送到网络192.19.12的数据包转发到IP地址为192.168.1.1的路由器。

Destination Gateway Genmask Flags Metric Ref Use Iface

---------- ------- ------- ----- ----- --- --- -----

192.19.12 192.168.1.1 255.255.255.0 UN 0 0 0 eth0

默认路由

当主机不能在路由表中查找到目标主机的IP地址或网络路由时，数据包就被发送到默认路由（默认网关）上。默认路由的Flags字段为G。例如，在下面的示例中，默认路由是IP地址为192.168.1.1的路由器。

Destination Gateway Genmask Flags Metric Ref Use Iface

---------- ------- ------- ----- ------ --- --- -----

default 192.168.1.1 0.0.0.0 UG 0 0 0 eth0

#添加删除路由

# route [add|del] [-net|-host] target [netmask Nm] [gw Gw] [[dev] If]

#查看ARP缓存

arp -n

Address HWtype HWaddress Flags Mask Iface

#查看网络

netstat -l [-n] [-p] [-e] [-t/u]; show tcp/udp listening => ss -lnrtp

netstat -r [-n]; pirnt route table => ip route

netstat -c -t/u [-n] [-p] [-e]; show tcp/udp connections; ctrl + c exit; =>ss -nrtp

netstat -s ; show statistics => ip -s link

netstat -i [-e]; show interfaces => ip a

lsof -i:<port>

#安装net-tools

apt-get install net-tools

#查看网络(iproute2)

ss -lntp

ss -ntp

#tcpdump

#Usage:（**Need root User**）

tcpdump [options] [expression]

#表达式：

#类型关键字

host

net

port

#方向关键字

dst

src

#协议关键字：

fddi，icmp，ip，arp，rarp，tcp，udp

#常用表达式

and &&

or ||

not !

命令参数：

#显示控制

-nn 显示portnumber和ip

#控制抓取数量

-c 100

#输出文件（默认输出控制台）

-w ./target.cap：保存成cap文件，方便用ethereal(即wireshark)分析

#输出详细信息（for控制台）

-v：当分析和打印的时候，产生详细的输出。

-vv：产生比-v更详细的输出。

-vvv：产生比-vv更详细的输出。

#输出数据

-X：输出包的头部数据，会以16进制和ASCII两种方式同时输出。

-XX：输出包的头部数据，会以16进制和ASCII两种方式同时输出，更详细。

#以ASCII码显示数据

-A 以ASCII码方式显示每一个数据包(不会显示数据包中链路层头部信息). 在抓取包含网页数据的数据包时, 可方便查看数据

#用户（如果报错No tcpdump user）

-Z user

#方向

-P：指定要抓取的包是流入还是流出的包。可以给定的值为"in"、"out"和"inout"，默认为"inout"。

#抓包长度

-s len：设置tcpdump的数据包抓取长度为len，如果不设置默认将会是65535字节。对于要抓取的数据包较大时，长度设置不够可能会产生包截断，若出现包截断，输出行中会出现"[|proto]"的标志(proto实际会显示为协议名)。但是抓取len越长，包的处理时间越长，并且会减少tcpdump可缓存的数据包的数量，从而会导致数据包的丢失，所以在能抓取我们想要的包的前提下，抓取长度越小越好。

#例子（**Need root User**）

#查看特定协议

tcpdump -p arp

#监控监听端口

tcpdump -vv port <listen-port>

#监控双向socket

tcpdump port <random-port>

#监控单向-接收

tcpdump src host 1.1.1.1 and src port <random-port>

#监控单向-接收

tcpdump dst host 1.1.1.1 and dst port <random-port>

#

tcpdump dst port <port> -c 100 -w ./a.cap

#random

正常情况下，Linux提供/dev/random（真随机）和/dev/urandom（强伪随机）两个设备供系统内应用使用，实际使用时因为/dev/random可能会卡住（随机数不足），基本上目前主流开源软件都使用/dev/urandom作为随机数来源，另：为了安全性和健壮性，需要引入自动补充随机数的软件havaged，作为常驻进程补充随机数。

#熵

Linux是根据系统的熵池来产生随机数的，熵池就是系统当前的环境噪音，环境噪音的来源很多，比如键盘的数据，鼠标的移动，内存的使用量，进程数等等。

查看系统熵池的容量：

cat /proc/sys/kernal/random/poolsize

查看熵池中拥有的熵数：

cat /proc/sys/kernal/random/entropy_avail

查看从熵池中读取熵的阀值，当entropy_avail中的值小于这个阀值，读取/dev/random会被阻塞。

cat /proc/sys/kernal/random/read_wakeup_threshold

通过开启haveged服务可以快速产生熵。

#随机数规范

要么使用真熵源的真随机数生成器，要么使用真熵源作为随机数种子，通过符合要求的伪随机数生成器产生随机数。

#生成base64随机字符串

cat /dev/random | head -c 16 | base64

openssl rand -base64 16

#生成hex随机字符串

openssl rand -hex 16

#监控入站链接

watch -n 1 'netstat -cnpt|grep 8080'

#useradd - create a new user or update default new user information

useradd [options] LOGIN

useradd -D

useradd -D [options]

#groupadd - create a new group

groupadd [options] group

#exaple of group user

groupadd sftp_group

useradd -g sftp_group -s /sbin/nologin sftp_user1

passwd sftp_user1

#chown

chown -R <owner[:group]> <path>

#crontab - maintain crontab files for individual users (Vixie Cron)

crontab  is the program used to install, deinstall or list the tables used to drive the cron(8) daemon in Vixie Cron.  Each user can have their own crontab, and though these are files in /var/spool/cron/crontabs, they are not intended to be edited directly.

crontab [ -u user ] file

crontab [ -u user ] [ -i ] { -e | -l | -r }

#start-stop-daemon - start and stop system daemon programs

#syslog日志

Jun 21 03:31:01 ip-10-10-2-214 CRON[30194]: (admin) CMD (/opt/apps/lms/data_import /import_data_to_micro_school.sh >> /var/log/lms/import_data_to_micro_school.log 2>&1)

Jun 21 03:31:01 ip-10-10-2-214 CRON[30193]: (CRON) info (No MTA installed, discarding output)

出现这个错误是由于admin用户没有log文件的写权限

#format of crontab file

minute hour day-of-month month-of-year day-of-week commands

第一项是分钟，第二项是小时，第三项是一个月的第几天，第四项是一年的第几个月，第五项是一周的星期几，第六项是要执行的命令。这些项都不能为空，必 须填入。如果用户不需要指定其中的几项，那么可以使用*代替。逗号(',') 指定列表值。如: “1,3,4,7,8"；中横线('-') 指定范围值 如 "1-6", 代表 “1,2,3,4,5,6"；星号 ('*') 代表所有可能的值。

#测试域名解析

dig <domain>

nslookup <domain>

#测试IP层网络联通性ping

ping uww-pro.thinkjoy.com.cn

#测试TCP层网络联通性telnet

telnet <ip|domain> <port>

telnet 10.200.249.10 3306

telnet 10.200.249.10 443

#测试HTTP或HTTPS连通性(尤其对定位HTTPS问题以及HTTP 2等协议问题)

curl -vvvv <web_url>

#了解进程io包括网络情况

lsof -p <pid>

#一个完整的HTTPS例子

```
curl -vvv https://www.163.com
* Trying 219.145.180.42...
* TCP_NODELAY set
* Connected to www.163.com (219.145.180.42) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
* CAfile: /etc/ssl/cert.pem
CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
* subject: C=CN; ST=Zhejiang; L=Hangzhou; O=NetEase (Hangzhou) Network Co.,
Ltd; OU=IT Dept.; CN=*.163.com
* start date: Feb 12 00:00:00 2020 GMT
* expire date: Apr 10 12:00:00 2022 GMT
* subjectAltName: host "www.163.com" matched cert's "*.163.com"
* issuer: C=US; O=DigiCert Inc; OU=www.digicert.com; CN=GeoTrust CN RSA CA
G1
* SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7fcfd9800600)
> GET / HTTP/2
> Host: www.163.com
> User-Agent: curl/7.64.1
> Accept: */*
```

>
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 403
< date: Wed, 30 Dec 2020 13:32:47 GMT
< content-type: text/html
< content-length: 233
< server: web cache
< expires: Wed, 30 Dec 2020 13:32:47 GMT
< x-ser: BC44_dx-shan3xi-weinan-3-cache-3
< cache-control: no-cache,no-store,private
< cdn-user-ip: 202.100.50.11
< cdn-ip: 219.145.180.42
< x-cache-remote: HIT
< cdn-source: baishan
<
<html><head><title>ERROR: ACCESS DENIED</title></head><body>
<h1>ERROR: ACCESS DENIED</h1> <hr>
Wed, 30 Dec 2020 13:32:47 GMT (taikoo/BC44_dx-shan3xi-weinan-3-cache-3) </BODY></HTML>
<!-- web cache -->
* Connection #0 to host www.163.com left intact
* Closing connection 0

#curl

--get:

curl <url> -vvvv

--get by query string:

curl <url?p1=aaa&p2=bbb> -v

--post by form:

curl -d “p1=aaa&p2=bbb” <url> -v

--head without host

curl --head --http1.0 -H 'Host:<host>' <url> -v

--resolve

curl --resove mywebsite.com:1.2.3.4 http://mywebsite.com/demo

curl [options...] <url>

curl  --user passwd:user XGET|XPUT url

curl --trace-ascii - <url>

#实例-正常网速

ping gitlib.com
PING gitlib.com (120.76.198.214) 56(84) bytes of data.
64 bytes from 120.76.198.214: icmp_seq=1 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=2 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=3 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=4 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=5 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=6 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=7 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=8 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=9 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=10 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=11 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=12 ttl=55 time=28.8 ms
64 bytes from 120.76.198.214: icmp_seq=13 ttl=55 time=28.7 ms
64 bytes from 120.76.198.214: icmp_seq=14 ttl=55 time=28.8 ms
^C
--- gitlib.com ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13024ms
rtt min/avg/max/mdev = 28.797/28.821/28.850/0.144 ms

PING gitlab.com (35.231.145.151): 56 data bytes
64 bytes from 35.231.145.151: icmp_seq=0 ttl=33 time=271.851 ms
64 bytes from 35.231.145.151: icmp_seq=1 ttl=33 time=274.456 ms
64 bytes from 35.231.145.151: icmp_seq=2 ttl=33 time=276.605 ms
64 bytes from 35.231.145.151: icmp_seq=3 ttl=33 time=272.556 ms
64 bytes from 35.231.145.151: icmp_seq=4 ttl=33 time=273.654 ms
64 bytes from 35.231.145.151: icmp_seq=5 ttl=33 time=458.874 ms
64 bytes from 35.231.145.151: icmp_seq=6 ttl=33 time=272.798 ms
64 bytes from 35.231.145.151: icmp_seq=7 ttl=33 time=276.719 ms
64 bytes from 35.231.145.151: icmp_seq=8 ttl=33 time=289.394 ms
^C
--- gitlab.com ping statistics ---
9 packets transmitted, 9 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 271.851/296.323/458.874/57.690 ms

#实例-异常网速
ping gitlab.com
PING gitlab.com (35.231.145.151) 56(84) bytes of data.
64 bytes from 151.145.231.35.bc.googleusercontent.com (35.231.145.151):

icmp_seq=7 ttl=36 time=357 ms
64 bytes from 151.145.231.35.bc.googleusercontent.com (35.231.145.151):
icmp_seq=19 ttl=36 time=356 ms
64 bytes from 151.145.231.35.bc.googleusercontent.com (35.231.145.151):
icmp_seq=27 ttl=36 time=357 ms
64 bytes from 151.145.231.35.bc.googleusercontent.com (35.231.145.151):
icmp_seq=33 ttl=36 time=357 ms
64 bytes from 151.145.231.35.bc.googleusercontent.com (35.231.145.151):
icmp_seq=49 ttl=36 time=350 ms
^C
--- gitlab.com ping statistics ---
58 packets transmitted, 5 received, 91% packet loss, time 57417ms
rtt min/avg/max/mdev = 350.141/355.886/357.575/2.957 ms

#DNS lookup
nslookup,dig
nslookup  www.163.com
#使用默认dns服务器查询
dig www.163.com
#使用指定的dns服务器查询
dig @4.2.2.2 www.163.com
#使用+trace参数可以显示DNS的整个分级查询过程。
dig +trace math.stackexchange.com
#单独查看每一级域名的NS记录。
dig ns com
dig ns stackexchange.com
#使用+short参数可以显示简化的结果。
dig +short ns com
dig +short ns stackexchange.com
#本地默认dns server配置
/etc/resolv.conf

#traceroute
traceroute <ip|host>
traceroute -P <tcp|udp|icmp> -p <port for udp/tcp> <ip|host>

#getip
curl http://ip.chinaz.com/getip.aspx
curl http://www.libvideo.com/getIp.aspx
curl http://myip.ipip.net

#查询google.com域名所有地址：

host -t a google.com

#ssh超时设置

客户端配置~/.ssh/ssh_config：

ServerAliveInterval 120

ServerAliveCountMax 3

服务器端配置/etc/ssh/sshd_config:

#服务端主动向客户端发送请求的间隔，默认为0，不发送。

ClientAliveInterval 120

#服务端发出请求后客户端没有响应的次数达到一定值就自动就自动断开

ClientAliveCountMax 3

#远程连接

ssh -vvvv

说明：

-i identity_file

Selects a file from which the identity (private key) for public key authentication is read. The default is ~/.ssh/identity for protocol version 1, and ~/.ssh/id_dsa, ~/.ssh/id_ecdsa, ~/.ssh/id_ed25519 and ~/.ssh/id_rsa for protocol version 2. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple -i options (and multiple identities specified in configuration files). If no certificates have been explicitly specified by the CertificateFile directive, ssh will also try to load certificate information from the filename obtained by appending -cert.pub to identity filenames.

-F configfile

Specifies an alternative per-user configuration file. If a configuration file is given on the command line, the system-wide configuration file (/etc/ssh/ssh_config) will be ignored. The default for the per-user configuration file is ~/.ssh/config.

connect:

ssh [-p port] user@host

execute:

ssh [-p port] user@host command

examples:

ssh root@10.32.0.190

ssh -p port root@10.32.0.190

ssh -o "ServerAliveInterval 120" root@10.32.0.190

ssh root@10.32.0.190 -i <privatekey>

ssh root@10.32.0.190 -F <cofig file>

ssh -o "IdentitiesOnly yes" -o "ProxyCommand ssh -W 192.168.32.48:22 -o IdentitiesOnly=yes -i ~/.ssh/wg/kwnc-wguser.key wguser@202.86.189.242" -i ~/.ssh/wg/kwnc-internal-wguser.key

wguser@192.168.32.48

ssh -o "IdentitiesOnly yes" -o "ProxyCommand ssh -W %h:%p -o IdentitiesOnly=yes -i ~/.ssh/wg/kwnc-wguser.key wguser@202.86.189.242" -i ~/.ssh/wg/kwnc-internal-wguser.key wguser@192.168.32.48

ssh-config:

说明：

使用范围：

1. command-line options
2. user's configuration file (~/.ssh/config)
3. system-wide configuration file (/etc/ssh/ssh_config)

ServerAliveCountMax

Sets the number of server alive messages which may be sent without ssh receiving any mes-sages back from the server. If this threshold is reached while server alive messages are being sent, ssh will disconnect from the server, terminating the session. It is important to note that the use of server alive messages is very different from TCPKeep (4.060, 0.110, 2.78%)Alive. The server alive messages are sent through the encrypted channel and therefore will not be spoofable. The TCP keepalive option enabled by TCPKeepAlive is spoofable. The server alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

The default value is 3. If, for example, ServerAliveInterval is set to 15 and ServerAliveCountMax is left at the default, if the server becomes unresponsive, ssh will disconnect after approximately 45 seconds.

ServerAliveInterval

Sets a timeout interval in seconds after which if no data has been received from the server, ssh will send a message through the encrypted channel to request a response from the server. The default is 0,indicating that these messages will not be sent to the server.

TCPKeepAlive

Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and some people find it annoying.

The default is yes (to send TCP keepalive messages), and the client will notice if the network goes down or the remote host dies. This is important in scripts, and many users want it too.

To disable TCP keepalive messages, the value should be set to no. See also ServerAliveInterval for pro-tocol-level keepalives.

config for normal:

Host 10.10.*

User admin

IdentitiesOnly yes

IdentityFile ~/.ssh/wg/aws-lms-prod.pem

ProxyCommand ssh -W %h:%p -i ~/.ssh/id_rsa daviszan@54.222.240.81

config for use multiple keys:

IdentityFile ~/.ssh/id_rsa

IdentityFile ~/.ssh/id_rsa_zxf

usage with config:

ssh -F ./ssh-config 10.10.2.214

usage without config:

ssh -o "IdentitiesOnly yes" -o "ProxyCommand ssh -W 10.10.2.214:22 -o IdentitiesOnly=yes -i ~/.ssh/id_rsa daviszan@54.222.240.81" -i ~/.ssh/wg/aws-lms-prod.pem admin@10.10.2.214

ssh -o "IdentitiesOnly yes" -o "ProxyCommand ssh -W %h:%p -o IdentitiesOnly=yes -i ~/.ssh/id_rsa daviszan@54.222.240.81" -i ~/.ssh/wg/aws-lms-prod.pem admin@10.10.2.214

ssh-keygen：这个程序可以生成密钥对。

ssh-keygen -t rsa -b 4096 -C “youremail@example.com”

//Delete item from known_hosts

ssh-keygen -f "/root/.ssh/known_hosts" -R 106.14.177.44

ssh-copy-id

install your public key in a remote machine's authorized_keys

ssh-copy-id [-i [identity_file]] [user@]machine

ssh-keyscan

ssh-agent：它把私钥保存在内存中，这样就不用反复输入口令了。

ssh-add：它把私钥加载到代理中。

sshd

#使用ssh-agent连接

eval $(ssh-agent)

ssh-add ~/.ssh/key-pair-40.pem

ssh -A root@100.114.100.154

ssh -o "IdentitiesOnly yes" -o "ProxyCommand ssh -W %h:%p -o IdentitiesOnly=yes -A root@202.86.189.242" -A root@192.168.32.48

#Decrypt an encrypted SSL key using OpenSSL

openssl rsa -in server.key.secure -out server.key

#远程复制(WinSCP)

scp <option> <source> <dest>

scp -vr root@10.32.0.190:/var/tmp/hbase-0.94.12.tar.gz ./

#其他网路命令

ftp，telnet，rlogin，finger，mail

#文本编辑器

vim(ESC：i,I, a,A, o,O, s,S)；

x，dw，d$(删到行尾)，dd(删单行)，ndd(删n行)，yy(拷单行)，nyy(拷n行)，p(粘贴)，u(撤销)，/xxxx(下搜，n->下一个，N->上一个)，?xxxx(上搜)

gg(文件开头)，G(文件末尾)，nG(到第n行)。

:sh，:wq，:q，:q!，:!，:!command，:<row_no>，:set nu，:set nonu，:% s/root/admin/g

:set ff --查看fileformat，:set ff=unix --设置fileformat

#进入shell命令行,执行完命令后ctrl+d退出重新进入vim编辑继续编辑

在shell命令下，执行ctrl+l完成清屏

#清屏

clear

#date 命令用于显示 / 设置系统的时间和日期，格式为“data [选项] [+指定的格式]”

date

date "+%H-%m-%d %H:%M:%S"

#确定用户身份

whoami, id

#查看系统当前其他用户: who - show who is logged on

who, who -a, who -H

#修改文件权限等信息

chmod [u,g,o]+[r,w,x] [file]

chmod u+x play

#文件的复制、删除、移动

cp，rm，mv；删除非空目录用rm -fr

#文件比较

diff，cmp

#diff && patch

只是文件夹比较

基于行比较，不支持二进制

#diff

diff -u a b > diff.txt

#patch from a to b

patch a < diff.txt

#patch from b to a

patch -R b < diff.txt

#Execute commands from a file in the current shell.

source filename [arguments]

Read and execute commands from FILENAME in the current shell. The entries in $PATH are used to find the directory containing FILENAME. If any ARGUMENTS are supplied, they become the positional parameters when FILENAME is executed.

Exit Status:

Returns the status of the last command executed in FILENAME; fails if FILENAME cannot be read.

#Replace the shell with the given command.

exec [-cl] [-a name] [command [arguments ...]] [redirection ...]

Execute COMMAND, replacing this shell with the specified program. ARGUMENTS become the arguments to COMMAND. If COMMAND is not specified, any redirections take effect in the current shell.

Options:

-a name pass NAME as the zeroth argument to COMMAND

-c execute COMMAND with an empty environment

-l place a dash in the zeroth argument to COMMAND

If the command cannot be executed, a non-interactive shell exits, unless the shell option `execfail' is set.

Exit Status:

Returns success unless COMMAND is not found or a redirection error occurs.

#创建Link

ln -s <source> <target>

ln -s /usr/share/play-2.1.0/play /usr/bin/play

#删除Link(删除Link切记不能使用-rf, 注意目录末尾不要带/，否则就是删除目录内容而不是link)

rm <link>

unlink <link>

#查找某个文件的link

find -L /dir/to/start -xtype l -samefile ~/Pictures

or, maybe better:

find -L /dir/to/start -xtype l -samefile ~/Pictures 2>/dev/null

#目录相关的命令：

cd，mkdir，rmdir，pwd

mkdir -p a/b/c/d/e

mkdir -p /etc/confd/{conf.d,templates}

#定时执行：

at，atq，atrm，batch

#检查文件内容的命令：

cat，touch，tac，nl，more，less，head，tail，od，vi，vim等

#touch 用于创建空白文件或修改文件时间，格式为“touch [选项] [文件]”

linux中的文件有三种时间：mtime - 内容修改时间，ctime - 权限或属性修改的时间，atime - 读取内容的时间

#使用cat增加hosts

cat <<EOF>> /etc/hosts

> 127.0.0.1  ccb.lms.com.cn

> EOF

#显示末尾的200行数据

tail -n200 [file] |less

#显示起始的200行

head -n200 [file] |more

#Socks Proxy: https://shadowsocks.org

#install

apt-get install python-pip

pip install shadowsocks

#run at foreground

ssserver -c /etc/shadowsocks.json

ssserver -p 443 -k password -m aes-256-cfb

#run as background

ssserver -c /etc/shadowsocks.json -d start

sudo ssserver -p 443 -k password -m aes-256-cfb --user nobody -d start

#to stop

sudo ssserver -d stop

ssserver -c /etc/shadowsocks.json -d stop

#check log

sudo less /var/log/shadowsocks.log

#config

```
{
    "server":"0.0.0.0",
    "server_port":8388,
    "local_address": "127.0.0.1",
    "local_port":1080,
    "password":"123987654",
    "timeout":300,
    "method":"aes-256-cfb",
    "fast_open": true
}
```

#显示请求和响应细节

ab -n 1 -v 4 -H "Accept-Encoding: gzip,deflate" http://101.132.140.243:8083/app.js

#解压解包文件 ( *.tar , *.tar.gz , *.tgz , *.gz , *.Z , *.bz2 )

tar -xzf [file.tar.gz]

tar -xjf [file.bz2]

tar -xf [file.tar]

打包压缩文件

tar -cf [file.tar] [folder|search pattern]

tar  - czf [file.tar.gz] [folder|search pattern]

tar  - cjf [file.bz2] [folder|search pattern]

#gzip

gzip -9 app.js

gzip -d app.js.gz

#压缩解压zip

zip , unzip

#zip

zip -r play.zip playbook

#unzip war

unzip abc.war

#下载文件

wget [url]

wget [url] -O [file]

#安装程序

apt-get install [name]

apt-get install -y [name]

#卸载程序

apt-get autoremove [name]

#升级程序

apt-get upgrade

#更新软件列表

apt-get update

#程序清单

apt-cache show package_name

apt-cache search package_name

apt-cache search trace

#编译程序

make

iotop

#常驻运行程序方式一

使用sysV或systemd的方式

#SysV系统启动相关文件：

init 系统启动超级进程

inittab 进程启动配置文件

rc0 - rc6 各启动级别的启动及停止脚本（S**代表启动，K**代表停止，全都link到init.d下的脚本）

init.d 启动脚本存放目录

#查看SysV服务

service --status-all

chkconfig

#SysV系统运行级别有0~6共7个运行级别：

0 - halt (Do NOT set initdefault to this)

停机（不能使用）

1 - Single user mode

单用户模式

2 - Multiuser, without NFS (The same as 3, if you do not have networking)

多用户模式，但是没有NFS

3 - Full multiuser mode

完全多用户模式

4 - unused

没有使用

5 - X11

图形界面模式

6 - reboot (Do NOT set initdefault to this)

重启模式（不能使用）

#使用init切换SysV的运行级别

init 3

## #systemd 配置文件介绍-缺失

#systemctl - control the systemd system and service manager

systemctl [options...] COMMAND [name...]

命令：

systemctl status [name...]

systemctl start [name...]

systemctl stop [name...]

systemctl reload [name...]

systemctl enable [name...]

systemctl disable [name...]

#to see services enabled on particular target

systemctl list-denpendencies [target]

#to list systemd services

systemctl list-unit-files

使用单元：

系统服务(.service)

挂载点(.mount)

Sockets(.socket)

系统设备(.device)

启动目标(.target)

计时器(.timer)

快照(.snapshot)

#交互式性能监视工具(top, htop,iftop)

top；1监控每个逻辑CPU；q 退出；P：按%CPU使用率排行；+H显示Thread；h：帮助命令。

htop

iftop

#cpu info

cat /proc/cpuinfo

#service - run a System V init script

service --status-all

service SCRIPT COMMAND [OPTIONS]

#关闭防火墙

service iptables stop

#查看防火墙状态

service iptables status

#iptables

iptables [-t table] {-A|-C|-D} chain rule-specification

iptables [-t table] -I chain [rulenum] rule-specification

iptables [-t table] -R chain rulenum rule-specification

iptables [-t table] -D chain [rulenum|rule-specification]

iptables [-t table] -S [chain [rulenum]]

iptables [-t table] {-F|-L|-Z} [chain [rulenum]] [options...]

iptables [-t table] -N chain

iptables [-t table] -X chain

iptables [-t table] -P chain target

iptables [-t table] -E old-chain-name new-chain-name

选项：

table：

-t, --table <table> 指定需要维护的防火墙规则表(filter，nat， raw，mangle，security)，不使用-t时，则

默认操作对象为filter表。

## commnd：

-A, --append，追加防火墙规则

-C, --check，检查

-D, --delete，删除防火墙规则

-I, --insert，插入防火墙规则

-R, --replace，替换防火墙规则

-L, --list，列出防火墙规则

-S, --list-rules，列出防火墙规则

-F, --flush，清空防火墙规则

-Z, --zero 清空防火墙数据表统计信息

-N, --new-chain 新建自定义链

-X, --delete-chain 删除自定义链

-E, --reanme-chain 重命名自定义链

-P, --policy 设置链默认规则

chain：

PREROUTING ( mangle , raw , nat )

FORWARD ( filter , mangle )

INPUT ( filter , mangle )

OUTPUT ( filter , raw , mangle , nat )

POSTROUTING ( nat , mangle )

rule-specification： CRETIRIA -j ACTION

匹配参数-CRETIRIA：！代表取反

[!]-p <...> 匹配协议

[!]-s <...> 匹配源地址

[!]-d <...> 匹配目标地址

[!]-i <...> 匹配入站网卡接口

[!]-o <...> 匹配出站网卡接口

[!]--sport <...> 匹配源端口

[!]--dport <...> 匹配目标端口

[!]--src-range <...> 匹配源地址范围

[!]--dst-range <...> 匹配目标地址范围

[!]--limit <...> 匹配数据表速率

[!]--mac-source <...> 匹配源MAC地址

[!]--sports <...> 匹配源端口

[!]--dports <...> 匹配目标端口

[!]--state <...> 匹配状态 ( INVALID , ESTABLISHED , NEW , RELATED )

[!]--string <...> 匹配应用层字串

触发动作-ACTION：

ACCEPT 允许数据包通过

DROP 丢弃数据包

REJECT 拒绝数据包通过

LOG 将数据包信息记录syslog日志

DNAT 目标地址转换 --to-source <new-src-ip>

SNAT 源地址转换 --to-destination <new-dest-ip>

MASQUERADE 地址欺骗

REDIRECT 重定向

自定义链：

```
#示例：在filter表中创建IN_WEB自定义链
iptables -t filter -N IN_WEB
#示例：在INPUT链中引用刚才创建的自定义链
iptables -t filter -I INPUT -p tcp --dport 80 -j IN_WEB
#示例：将IN_WEB自定义链重命名为WEB
iptables -E IN_WEB WEB
#示例：删除引用计数为0并且不包含任何规则的WEB链
iptables -X WEB
```

#iptables 配置持久化以及备份与恢复

对iptables防火墙规则的修改会立即生效，但如果不保存则系统下次重启后这些规则就会丢失。CentOS 6.3系统中防火墙的规则默认保存在/etc/sysconfig/iptables文件中。

```
#规则持久化
iptables-save > /etc/sysconfig/iptables
#备份与恢复
iptables-save > firewall.bak
iptables-restore  < firewall.bak

#查看filter表的所有规则
iptables -nL

#查看nat表的所有规则
iptables -t nat -nL

#清空filter表的所有规则
iptables -F

#禁止及开启Linux对某IP的路由转发
iptables - A FORWARD -s 192.168.199.235 -j REJECT
```

iptables - D FORWARD -s 192.168.199.235 -j REJECT

#su - change user ID or become superuser

[sodu] su

su <user>

#网络设备配置

/etc/sysconfig/network-scripts

#netns

/var/run/netns

/var/run/docker/netns

#网路设备信息

#网络设备别名

/sys/class/net/<dev>/ifalias

#网络设备编号

/sys/class/net/<dev>/ifindex

#网路设备配对

/sys/class/net/<dev>/iflink

/sys/class/net/<dev>/link_mode

#网桥设备接口

/sys/class/net/<dev>/brif/*

#接口所属网桥及port

/sys/class/net/<dev>/brport

/sys/class/net/<dev>/master

#网桥管理命令

#创建 Bridge :

brctl addbr [BRIDGE NAME]

#删除 Bridge :

brctl delbr [BRIDGE NAME]

#attach 设备到 Bridge :

brctl addif [BRIDGE NAME] [DEVICE NAME]

#从 Bridge detach 设备 :

brctl delif [BRIDGE NAME] [DEVICE NAME]

#查询 Bridge 情况 :

brctl show

#查看所有bridge

ip link show type bridge

#查看某个bridge的接口

bridge link show docker0

#vlan管理

#创建 VLAN 设备：

vconfig add [PARENT DEVICE NAME] [VLAN ID]

#删除 VLAN 设备：

vconfig rem [VLAN DEVICE NAME]

#设置 VLAN 设备 flag：

vconfig set_flag [VLAN DEVICE NAME] [FLAG] [VALUE]

#设置 VLAN 设备 qos：

vconfig set_egress_map [VLAN DEVICE NAME] [SKB_PRIORITY] [VLAN_QOS]

vconfig set_ingress_map [VLAN DEVICE NAME] [SKB_PRIORITY] [VLAN_QOS]

#查询 VLAN 设备情况：

cat /proc/net/vlan/[VLAN DEVICE NAME]

#veth

创建 VETH 设备：ip link add link [DEVICE NAME] type veth

#查看veth

ip link show type veth

#tap

创建 TAP 设备：tunctl -p [TAP DEVICE NAME]

删除 TAP 设备：tunctl -d [TAP DEVICE NAME]

#dev

查询系统里所有二层设备，包括 VETH/TAP 设备：ip link show

删除普通二层设备：ip link delete [DEVICE NAME] type [TYPE]

#查看更改网络接口信息

ifconfig，ip

#ip

#查看ip信息

ip a

#Set ip for nic

ip addr change [ip]/[mask-digits] dev [nic]

#Add second ip for nic

ip addr add [ip]/[mask-digits] dev [nic]

例子：

ip addr add 192.168.99.37/24 dev eth0

#Delete ip for nic

ip addr del 192.168.99.37/24 dev eth0

#启用网卡

ifconfig [<nic>] up

例子：

ifconfig eth0 up

#停用网卡

ifconfig [<nic>] down

例子：

ifconfig eth0 down

#停用网卡第二IP，即IP Alias

ifconfig [<nic>][:<tag>] down

例子：

ifconfig eth0:0 down

#设置网卡默认ip并启用

ifconfig [nic] [IP-Address] netmask [mask] up

例子：

ifconfig eth0 192.168.1.2 netmask 255.255.255.0 up

#设置网卡第二个IP，即IP Alias，并启用

ifconfig [nic]:[tag] [IP-Address] netmask [mask] up

例子：

ifconfig eth0:0 192.168.1.3 netmask 255.255.255.0 up

#dhcp获取地址

dhclient eth0

#任务项目命令

fg、bg、jobs、&、ctrl + z

：& 最经常被用到

这个用在一个命令的最后，可以把这个命令放到后台执行

：ctrl + z

可以将一个正在前台执行的命令放到后台，并且暂停

：jobs

查看当前有多少在后台运行的命令

：fg [##]

将后台中的命令调至前台继续运行

如果后台中有多个命令，可以用 fg %jobnumber将选中的命令调出，%jobnumber是通过jobs命令查到的后台正在执行的命令的序号(不是pid)

：bg

将一个在后台暂停的命令，变成继续执行

如果后台中有多个命令，可以用bg %jobnumber将选中的命令调出，%jobnumber是通过jobs命令查到的后台正在执行的命令的序号(不是pid)

blkid 显示块设备属性信息:
blkid

Fstab :
/etc/fstab
man fstab

#列出磁盘上的所有分区
fdisk -l

#系统中每个磁盘分区的主辅设备号、大小和名称
cat /proc/partitions

#当前内核所知的文件系统
cat /proc/filesystems

Mount(mount -t <type> <device> <dir>)

mount -t cifs //10.32.0.130/Share /media/share -o

username=******,password=*****,workgroup=grapecity.net

mount -t iso9660 /dev/cdrom /mnt

mount -t ext4 /dev/xvde /var/lib/go-server

Umount(uount <dir>)

umount //10.32.0.130/Share

umount /dev/cdrom

umount /dev/xvde

rpm:

rpm -ivh jdk-7u7-linux-x64.rpm

wget:

wget http://mirror.bit.edu.cn/apache/hadoop/common/hadoop-2.0.6-alpha/hadoop-2.0.6-alpha.tar.gz