1. shell是实现Linux系统自动化管理的重要且必要的工具，是一个命令解释器，位于操作系统的最外层，负责将解释命令，然后将处理结果返回给屏幕。学好shell的基础：
   1. vim的熟练使用
   2. Linux150个常用命令的熟练使用
   3. 常见Linux网络服务部署及排错
2. 层次：硬件=>操作系统内核=>shell=>外围应用程序，什么是shell脚本：当Linux命令不在命令行下执行，而是在文件中执行的时候，此时就称之为shell脚本。Windows的脚本则是以.bat（批处理文件）结尾的。make && make install如果前面的执行成功，后面的才会执行。demsg在系统宕机的时候查看系统信息的命令。
3. Shell在运维工作的作用和地位：shell擅长处理纯文本类型的数据，Linux系统中几乎所有的系统配置文件都是纯文本文件，所以shell编程是运维的必要且最重要的工具。
4. Shell脚本语言的种类：包括sh ksh bash csh tcsh，shell是一种若类型语言，较为通用的shell类型就是bash了，其他运维人员常用的脚本语言：PHP（专注于web页面的开发）、perl（前几年很火，比shell强大的多，但是语法太灵活，不易读，团队协作比较困难）、Python（比较火，可以做web开发，也可以做脚本开发）。四中脚本语言的区别于优势：shell相比于这几种语言，优势在于处理操作系统底层的业务，因为有大量的系统为其服务。PHP于Python优势在于开发高性能的运维工具、web界面的管理工具以及web业务的开发等，处理一键软件安装、优化、报警脚本就显得有点鸡肋了，开发难度也比较大，反正shell就是短小精悍。
5. Shell脚本的建立和执行：通常在vim编辑器中编写，脚本的第一行一般以#!/bin/bash或者#!/bin/sh，其作用就是指出由哪个程序来解释程序，必须位于脚本的第一行，如果不是第一行，则为脚本注释行，如果不写，则用Linux系统默认的解释器进行执行。系统脚本一般都是…./bash，而应用程序的脚本一般是…./sh。脚本执行的时候，一般会先查找系统变量，如果做crond任务的时候，一定要把环境变量重新定义。Shell脚本执行之前，一般会先读.bashrc,.bash,/etc/profile,/etc/bashrc,.bash\_profile这些文件中的内容，因为这些文件存放系统全局变量。执行方式：bash或这sh开头，全路径执行，但是必须要有可执行权限，source script-name或则. script-name（注意一定要有空格）,这种执行方式，可以把这个文件中定义的变量带到shell环境中（！重要），但是当用户退出系统的时候，这个变量就会在内存中消失。Shell脚本可以自定义脚本。
6. Shell脚本开发的基本规范和习惯：开头指定脚本解释器，开头加版本版权信息等内容，可以修改-/.vimrc配置文件自动添加，脚本以.sh结尾。中括号两边要有空格，书写的时候就把空格加上，引号要陈双写，成对的内容要一次写出来，流程控制语句一次性写完，然后添加内容，通过缩进让代码易读。
7. 变量基础及深入：一个固定的字符串，替代更多更复杂的内容。变量分为环境变量和局部变量。全局变量可以在创建他们的shell及其派生出来的任意子进程中使用。
8. 环境变量：用于定义shell运行环境，保证shell命令的正确运行。在命令行中定义的全局变量当用户退出的时候就会失效，所以应该在配置文件中定义全局变量。所有的环境变量都是全局的，不用定义可以直接使用环境变量。所有的环境都是大写，这是规范，不是必须的。环境变量的定义一般都用export导出。查看系统环境变量的命令：env和set命令都可以显示一些环境变量。
9. 自定义环境变量：export 变量名=value，如果不是在配置文件中，用户退出的时候就会失效，在.bash\_profile定义此变量就可以。
10. 环境设置的常用的文件及区别：用户环境变量的定义在.bash\_profile中，系统环境变量/etc/profile中定义。用户登陆之后需要加载哪些东西，可以在/etc/profile.d中写一下脚本，会在用户登陆之后执行也可以在/etc/motd中定义。
11. 显示环境变量：echo $HOME printf $HOME，取消变量名：unset 变量名，没有$符号。
12. 局部变量：本地变量在用户当前的shell生存期的脚本中使用。
13. 普通字符串变量定义：变量名一般是字母、数字、下划线组成，字母开头，支持变量插值定义。变量定义习惯，数字不加引号，其他都加双引号。
    1. 变量名=value
    2. 变量名=‘value’单引号原样输出，即使有变量，也会把变量名原样的输出。
    3. 变量名=“value”支持插值变量，即将内容中的变量取得其值，然后字符串拼接，如果有命令的话就用反引号进行转义，也可以用$(commond)进行执行。
14. 单引号双引号区别：单引号所见即所得。建议：除了连续的数字，其他的都是双引号。
15. 变量定义的方式：单词之间用下划线隔开。
16. Shell的特殊变量：
    1. 位置变量：$0 获取当前执行的shell脚本的文件名，包括脚本路径。
    2. $n 获取当前执行的shell脚本第n个参数值，n=1..9，大于9的话，要用{}括起来 sh test.sh sh test.sh syge 得到syge
    3. $# 获取当前脚本命令行中参数的个数
    4. $? 获取上一个命令的返回值，返回0表示执行正确，其他的返回值都是错误。如返回2则表示是没有权限执行该命令，未找到命令则返回127
    5. 脚本退出的时候exit 1，表示命令结束的时候返回1，就是将1赋值给$?了。
17. 系统脚本：执行命令的时候后面加一个参数，就会被$1获取到。
18. $\*和$@的区别：了解即可，$\*是获取所有shell的所有参数，把所有的参数当做一个字符串，而$@是将每一个参数当做一个字符串。
19. Shift：移动变量位置
20. $$:当前shell的进程号。可以应用于一个系统中只允许运行一个脚本实例
21. Bash的内置命令：常用的bash命令有可以man cd就可以查看有哪些内置命令，常用的有alias、pwd、printf、shift、echo、exec、set、type、unset、unalias、help、history、continue、logout、cd、ulimite、eval、export
22. 变量子串的常用操作：${#string}返回$string的长度.echo ${username}|wc –L这样也已得到变量的长度。${username:2}：从第二个位置开始取值。${username:2:4}:从第二个截取到第四个。管道命令也可以截取字符串：cut –c，类似于Java中的处理字符串的函数，这个了解即可。
23. 写脚本删除文件的时候一定要判断这个文件目录是否存在，如果不判断的话可能会从根目录开始删除。
24. 变量的计算常用命令：
    1. (()) （这个比较常用，效率最高，建议使用）如：echo $((a++))
    2. let let I = I + 8 如果不加let，那么他只是一个赋值运算罢了。
    3. expr：用来求表达式的值，运算符左右都要有空格，如果没有的话会有语法错误，计算乘法的时候要进行转义。expr “” : “”，可以用来判断文件的扩展名。
    4. bc（可以计算小数，其他的只可以计算整数），就是一个计算器，bc命令即可
    5. $[]：可以直接进行计算。
25. read命令：为内置命令，常用的参数有-p （提示信息）-t（超时时间）例如：

read –t 5 –p “please input:” a(a是一个变量，也可以输入多个变量，然后用空格隔开)

26、&&和||：用于判断前一个命令的回传值对于后一个命令是否要进行的依据。

27、test：可以用作判断，可以判断文件目录是否存在以及对应是否有权限，还可以判断字符串是否是空字符串。

28、[]：中括号也可以用作判断，例如：[ -z $HOME ];echo $? 注意中括号两侧以及判断表达式的两侧一定要加上空格。即中括号内的每个组件都需要用空格隔开，中括号内的变量最后都要用双引号括号起来，在中括号内的常量，最好都以单或者是双引号括号起来，

29、shift：移动变量的意思，如果有两个变量，移动一次，对应的将所有的变量的位置向左移动一位。

30、判断表达式：if then 格式：if [ expr ];then

elif [ expr ];then

else

fi

如果有多个判断表达式，则可以将多个表达式写在中括号内，然后用逻辑判断符进行判断，即&&和||

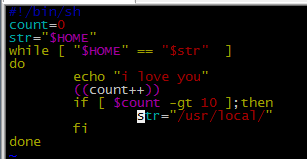
31、function函数：shell脚本执行的顺序为从上到下，从左到右，所以引用函数之前必须要先定义函数，语法：function myfunction( ){

}

32、循环表达式：while [ expr ]

do

done

例如：

33、当条件满足的时候停止循环：

Until [ ]

Do

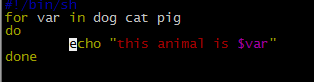
Done

34、固定次数的循环：

语法： for var in 一堆变量

do

done

例如：

35、固定循环的数值处理：

for (( 初始值; 限制值; 执行步长 ))

do

done

36、shell脚本调试：sh -n test.sh，检查语法