## 你能讲讲Shell编程吗？

Shell脚本语言是Linux/Unix系统上非常重要的脚本语言。它可以是一些简单的Linux命令的集合，可以定义变量，也可以编写函数、流程语句（if、switch）、表示式等。在开发和调试过程中，把一些经常使用到的Linux命令写入到一个shell脚本中，以后只要拍通过./sh来执行shell脚本，可以提高开始效率。

## Shell编程的应用：

1. 在开发和调试工程中，把一些经常使用到的Linux命令写入一个shell脚本，以后只要执行脚本，提高开发效率。

## Shell脚本

1. Shell有两种执行命令的方式：
2. 交互式：用户输入一条，Shell就解释执行一条
3. 批处理式：用户实现编写一个Shell脚本，其中有很多条命令，让Shell一次把这些命令执行完。

2、运行shell脚本 ./test.sh 告诉系统，在当前目录查找shell脚本运行。

## 基础语法：

1、#! 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种shell。比如，#!/bin/sh 标明使用bash解析器解析当前的脚本。

2、定义变量：

my\_sum=“dadad” //注意！！ 这里=左右不能有空格

echo ${my\_sum}

使用变量时，才用$

1. 只读变量: 使用readonly可以将变量声明为只读变量，只读变量的值不能改变

myUrl="http://see.xidian.edu.cn/cpp/shell/"

readonly myUrl

1. 删除变量：unset可以删除变量

#!/bin/sh

myUrl="http://see.xidian.edu.cn/cpp/u/xitong/"

unset myUrl

echo $myUrl

1. 运行Shell时，存在3种变量：
2. 局部变量：在脚本或者命令中定义，只在当前shell实例中有效，其他shell启动的程序不能访问局部变量。
3. 环境变量：所有程序，包括shell启动的程序，都可以访问环境变量。
4. shell变量：shell程序设置的特殊变量。
5. $代表当前进程的ID，利用echo $$可以输出当前进程的进程ID
6. 命令替换：Shell可以先执行命令，保存结果，在适当的时候输出。

#!/bin/bash

DATE=`date` //这里date是一个指令，date获取的值存到DATE中,命令用``符号括起来

echo "Date is $DATE"

USERS=`who | wc -l`

echo "Logged in user are $USERS"

UP=`date ; uptime`

echo "Uptime is $UP"

1. expr 表达式计算工具

able=`expr 1 + 1` // 注意！ +号左右需要有空格

1. if表达式

if [ 表达式 ]

then

执行的语句

fi

9.0、定义函数: function function\_name() {

[line of command]

Return value;

}

1. 获得上一个函数的返回值：
2. 通过$?获得 （2）把一个函数的返回值存在一个全局变量中

10、在shell中包含另外一个shell文件 . 文件的相对路径及文件名

## 经典题型：

1. 统计出现次数最多的前3个ip及其次数？

sort ip.txt | uniq -c | sort -rn | head -n 3

sort 会让ip进行排序，相同的ip排在一起；

uniq –c 去掉重复的ip，并且把该ip出现次数放在行首

sort –rn 按照行首的数字进行反向排序 –n是按照数字 -r是降序

head –n 3 取出前面3行

1. 查找当前目录及子目录下所有扩展名为”.conf”的文件

find . –name “\*.conf”

1. 对单个文件进行文本替换：

sed –i ‘/s/要替换的内容/替换成内容/g’ 替换的文件路径

sed –I ‘/s/One/Haha/g’ test.txt

// 注意，如果是在Mac上，要在-i前面加上’.bak’

1. 对文件内所有数字求和

awk 对文件逐行读入，以空格为默认分隔符，进行切片，切开的部分进行分析处理。 awk ‘{pattern + action}’ {filenames}

$1 获取第一个参数

实例： awk 'sum += $1 END {print sum}' test.txt

## 零碎知识：

1、编译性语言和解释型语言（脚本语言）：

（1）编译型语言：比如C++、Java，每次运行程序，直接读取目标代码，因为目标代码接近计算机底层，所以运行效率非常高。

（2）解释型语言：执行这类程序时，每次都要把程序编译目标代码，再执行，因此效率有所下降。