第一章：让工作变得高效率的shell介绍

简介：分三节课，介绍什么是shell; 还有应用场景跟大纲;基本操作命令

1.1 什么是shell？shell的简介介绍

简介：详细介绍shell

> Shell是一种脚本语言，又是一种命令语言。可以通俗一点来讲，Shell脚本就是一系列命令的集合，可以在Unix/linux上面直接使用，并且直接调用大量系统内部的功能来解释执行程序把一些重复性工作交给shell做，来实现自动化运维。Shell 虽然没有C/C++、Java、Python等强大，但也支持了基本的编程元素。例如：if、for、while等循环，还有变量、数组、字符串、注释、加减乘除逻辑运算等

- 适合人群：

> 这套视频教程面向任何对linux平台感兴趣的人，linux初学者，shell零基础都可以学。运维工程师、网络管理员、程序开发人员 更应该学习shell（尤其是linux运维工程师，对shell学习必不可少，必须掌握的技能）

- 学后水平：

> 可以写一些常用的小工具，小脚本来实现想要的功能，实现自动化运维

1.2 shell的应用场景以及课程大纲

简介：详细的介绍本套课程的大纲以及shell的应用场景

- 应用场景：

> Shell 主要用来开发一些实用的、自动化的小工具，而不是用来开发具有复杂业务逻辑的中大型软件
>
> 第一方面：监控linux系统环境的使用情况
> 第二方面：数据的处理。eg：日志的切割、分析、统计等
> 第三方面：与数据库的交互，对数据库进行增，删，改，查等操作
> 第四方面：监控进程，自动化启停服务进程
> 第五方面：完成一些重复性的工作。eg：创建100个新用户;到服务器集群配置某个文件等

- 课程大纲：总共七章

> 第一章：让工作变得高效率的shell介绍
> 1.1 什么是shell？shell的简介介绍

```
    1.2 shell的应用场景以及课程大纲
    1.3 linux的基本操作命令

第二章：shell的使用技巧
    2.1 推荐使用的远程连接软件以及vi编辑器的基本使用
    2.2 linux必备基础知识之shell常见的解释器
    2.3 linux必备基础知识之shell脚本文件权限与脚本执行

第三章：shell的变量以及常见符号
    3.1 常见变量
    3.2 常用的几个符号
    3.3 秒变计算器的运算符
    3.4 常见的条件判断

第四章：shell脚本的输入以及脚本拥有特效地输出
    4.1 shell脚本输入之read命令
    4.2 shell脚本输出上色

第五章：处理海量数据的grep、cut、awk、sed 命令
    5.1 处理海量数据之grep命令
    5.2 处理海量数据之cut命令
    5.3 处理海量数据之awk命令
    5.4 处理海量数据之sed命令

第六章：神奇的循环控制语句if 、for、case、while
    6.1 if 循环控制（单分支与多分支）
    6.2 for循环控制
    6.3 case循环控制
    6.4 while 循环

第七章：shell的几个实战脚本例子
    7.1 如何让shell实现 可选择性执行 的功能
    7.2 巡检内存使用率
    7.3 批量创建用户
    7.4 数据库里查询学生成绩
    7.5 如何实现高效率登录别的机器
```

1.3 linux的基本操作命令

简介：30个常用的命令

```
1、  cd命令
功能说明：切换目录。
举    例：却换到根目录 ：cd /

2、  ls命令
功能说明：列出目录内容。
举    例：列出/var目录的文件和目录的信息 ：ls -l /var；最常用方式 ls -ltr

3、  cat命令
功能说明：查看小文件内容。
举    例：查看test.txt 文件内容 ：cat test.txt
```

4、 chmod命令
功能说明：修改文件或目录权限。
举 例：修改test.sh 为自己可执行：chmod u+x test.sh

5、 chown命令
功能说明：变更文件或目录的拥有者或所属群组。
举 例：修改test.txt 属主为mysql ：chown mysql:mysql test.txt

6、 cp命令
功能说明：拷贝文件。
举 例：拷贝文件test.sh 为 test.sh_bak：cp test.sh test.sh_bak

7、 diff命令
功能说明：对比文件差异。
举 例：对比文件test.sh test.sh_bak 是否有差异diff test.sh test.sh_bak

8、 find命令
功能说明：查询文件。
举 例：查询本目录下面的test.txt：find ./ -name test.txt

9、 mv命令
功能说明：移动或更名现有的文件或目录。
举 例：移动 test.sh到/bin目录下：mv test.sh /bin/

10、rm命令
功能说明：删除文件或目录。
举 例：删除文件test.sh ：rm test.sh

11、touch命令
功能说明：创建一个空文件。
举 例：创建一个空的test.txt文件：touch test.txt

12、which命令
功能说明：在环境变量$PATH设置的目录里查找符合条件的文件。
举 例：查询find命令在那个目录下面：which find

13、ssh命令
功能说明：远程安全登录方式。
举 例：登录到远程主机：ssh ${IP}

14、grep命令
功能说明：查找文件里符合条件的字符串。
举 例：从test.txt文件中查询test的内容：grep test test.txt

15、wc命令
功能说明：统计行。
举 例：统计test.txt文件有多少行：wc -l test.txt

16、date命令
功能说明：查询主机当前时间。
举 例：查询主机当前时间：date

17、exit命令

功能说明：退出命令。
举　　例：退出主机登录：exit

18、kill命令
功能说明：杀进程。
举　　例：杀掉test用户下面的所有进程：ps -ef | awk '$1=="test" {print $2}' | xargs kill -9

19、id命令
功能说明：查看用户。
举　　例：查看当前用户：id ；查询主机是否有test用户：id test

20、ps命令
功能说明：查询进程情况。
举　　例：查询test.sh进程：ps -ef | grep test.sh

21、sleep命令
功能说明：休眠时间。
举　　例：休眠60秒 ：sleep 60

22、uname命令
功能说明：查询主机信息。
举　　例：查询主机信息：uname -a

23、passwd命令
功能说明：修改用户密码。
举　　例：使用root修改test用户的密码：passwd test

24、ping命令
功能说明：查看网络是否通。
举　　例：查询本主机到远程IP的网络是否通：ping ${IP}

25、df命令
功能说明：查看磁盘空间使用情况。
举　　例：查看主机的空间使用情况 ：df -h

26、echo命令
功能说明：标准输出命令。
举　　例：对变量test进行输出：echo $test

27、pwd命令
功能说明：查询所在目录。
举　　例：查询当前所在目录：pwd

28、head命令
功能说明：查看文件的前面N行。
举　　例：查看test.txt的前10行：head -10 test.txt

29、tail命令
功能说明：查看文件的后面N行。
举　　例：查看test.txt的后10行：tail -10 test.txt

30、mkdir命令

```
功能说明：创建目录。
举      例：创建test目录：mkdir test
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第二章：shell的使用技巧

简介：分为3节课

```
第一节：远程软件跟如何使用vi编辑器。
第二节：shell的解释器
第三节：shell脚本是怎么来执行的
```

2.1 推荐使用的远程连接软件以及vi编辑器的基本使用

简介:远程连接软件 与 vi命令的基本使用

- 软件：

```
    CRT
    putty
    echo $LANG
```

- vi编辑器：

```
vi编辑器是所有unix及Linux系统下标准的编辑器，它的强大不逊色于任何最新的文本编辑器
```

- vi的基本概念：

```
1）命令行模式
x               # 删除一个字符
dd              # 删除一整行

2）插入模式
i               # 在光标前插入内容
o               # 在当前行之下新开一行

3）底行模式
x 或 wq        # 保存退出
q!              # 退出不保存
set nu          # 显示行数
/               # 搜索内容
```

2.2 linux必备基础知识之shell常见的解释器

简介：常见的解释器

解释器:是一种命令解释器，主要作用是对命令进行运行和解释，将需要执行的操作传递给操作系统内核并执行

```
# !/bin/bash（默认）

# !/bin/ksh

# !/bin/bsh

# !/bin/sh
```

注意：面试官，shell一定得有解释器吗？
答案：不一定

- 课堂脚本：

```
#!/bin/bash
# 这是我的第一个shell脚本
# by 小C 2018年12月
echo 'this is my first shell ！！！'
```

2.3 linux必备基础知识之shell脚本文件权限与脚本执行

简介：shell是怎么执行的

```
文件权限：- rw-  r-- r--
目录权限：d rw-  r-- r--
分三列：每三个为一列，分别是所有者(owner)，所属组(group)，其他(others)

rwx r:4 w:2 x:1

7   5   5
```

- 执行方法：

```
    方法1：添加执行权限 chmod +x shell.sh

    方法1：./shell.sh

    方法2：sh shell.sh 或者bash shell.sh

    方法3：source shell.sh
```

第三章：shell的变量以及常见符号

简介：分为四节课

```
第一节：变量
第二节：常见符号
第三节：常见的运算符
第四节：常见的条件判断
```

3.1、常见变量

```
不同于其它语言需要先声明变量
shell的变量直接使用，eg:a=15
调用变量的话 $或者a 或者 ${a}

$?   #判断上一条命令执行的是否成功

$0   #返回脚本的文件名称

$1-$9 #返回对应的参数值

$* #返回所有的参数值是什么

$# #返回参数的个数和
```

- 课堂脚本：

```
#!/bin/bash
# by xiao C 2018-12
# test
echo "脚本：$第一个参数是：0"
echo "第一个参数是：$1"
echo "第二个参数是：$一共有多少参数2"
echo "一共有多少参数:$#"
echo "这些参数是什么：$*"
```

3.2、常见的几个符号

```
>              #会覆盖原有的内容

>>             #不会覆盖原有的内容
```

```
;               #执行多条命令

|               #管道符

&&              #前面的命令执行成功，后面的才可以执行

||              #前面的命令执行失败，后面的才可以执行

""              #会输出变量值

''              #输出本身

``              #输出命令结果 eg:a=`date`;echo $a

2>/dev/null   #错误输出到无底洞
1>/dev/null   #正确输出到无底洞
```

3.3、秒变计算器的运算符

- 整数:

```
加：expr 12 + 6       expr $a + $b

    echo $[12 + 6]      echo $[a + b]

    echo $((12 + 6))    echo $((a + b))

减：expr 12 - 6          expr $a - $b

    echo $[12 - 6]      echo $[a - b]

    echo $((12 - 6))    echo $((a - b))

乘：expr 12 \* 6      expr $a \* $b

    echo $[12 * 6]      echo $[a * b]

    echo $((12 * 6))    echo $((a * b))

除：expr 12 / 6          expr $a / $b

    echo $((12 / 6))    echo $((a / b))

    echo $[12 / 6]      echo $[a / b]

求余：expr 12 % 6      expr $a % $b

      echo $((12 % 6))  echo $((a % b))

      echo $[12 % 6]    echo $[a % b]
```

- 小数：

```
            bc计算器
            保留多少位小数可以通过scale
            但是scale只对除法，取余数，乘幂 有效，对加减没有效。

            echo "scale=2;(0.2+0.3)/1" | bc    #计算出0.2+0.3的和并保留俩位小数，此时bc计算器会省略掉
个位数的0
            echo "scale=2;(1.2+1.3)/1" | bc    #计算出1.2+1.3的和并保留俩位小数
```

3.4、常见的条件判断

- 语法：[判断表达式]

```
文件（夹）或者路径：
-e  目标是否存在（exist）
-d  是否为路径（directory）
-f  是否为文件（file）
[ -e foer.sh ] || touch foer.sh      #判断当前目录下是否有foer.sh这个文件，假如没有就创建出foer.sh文
件
```

- 权限：

```
-r  是否有读取权限（read）
-w  是否有写入权限（write）
-x  是否有执行权限（excute）

[ -x 123.txt ] && echo '有执行权限'
```

- 整数值（int型）：

```
-eq 等于（equal）
-ne 不等于(not equal)
-gt 大于（greater than）
-lt 小于（lesser than）
-ge 大于或者等于（greater or equal）
-le 小于或者等于（lesser or equal）

[ 9 -gt 8 ] && echo '大于'
```

- 小数（浮点型）：

```
[ `echo '1.2 < 1.3' | bc` -eq 1 ] && echo '小于'
```

- 字符串：

```
=     相等
!=    不相等
[ 'kkkkk' != 'kkkk' ] && echo '不等于'
```

- 课堂脚本1：

```
#!/bin/bash
# 判断输入的第一个数是否大于输入的第二个数
# by 小C

if [ $1 -eq $2 ]
then
echo "$等于1 等于 $2"
else
echo "$不等于1 不等于 $2"
fi
```

- 课堂脚本2：

```
#!/bin/bash
touch $1
if [ $? -eq 0 ];then
echo "$1 创建成功！"
fi
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第四章：shell脚本的输入以及脚本拥有特效地输出

简介：分俩节课

```
第一节：用于读取终端输入的read命令
第二节：给脚本输出上色
```

4.1、shell脚本输入之read命令

- 语法：read -参数

```
-p：给出提示符。默认不支持"\n"换行
-s：隐藏输入的内容
-t：给出等待的时间，超时会退出read
-n：限制读取字符的个数，触发到临界值会自动执行
```

- 课堂脚本:

```
#!/bin/bash

read -p "请输入您的密码：" pass
echo $pass
```

```
#!/bin/bash

echo "请您输入密码："
read pass
echo "你输入的密码是:$pass"
```

4.2、shell脚本输出上色

- 语法: echo -e "\033[字背景颜色;字体颜色;特效字符串\033[关闭属性"

```
    #字体色范围：30-37
        echo -e "\033[30m 黑色字 \033[0m"
        echo -e "\033[31m 红色字 \033[0m"
        echo -e "\033[32m 绿色字 \033[0m"
        echo -e "\033[33m 黄色字 \033[0m"
        echo -e "\033[34m 蓝色字 \033[0m"
        echo -e "\033[35m 紫色字 \033[0m"
        echo -e "\033[36m 天蓝字 \033[0m"
        echo -e "\033[37m 白色字 \033[0m"
    #字背景颜色范围：40-47
        echo -e "\033[40;37m 黑底白字 \033[0m"
        echo -e "\033[41;30m 红底黑字 \033[0m"
        echo -e "\033[42;34m 绿底蓝字 \033[0m"
        echo -e "\033[43;34m 黄底蓝字 \033[0m"
        echo -e "\033[44;30m 蓝底黑字 \033[0m"
        echo -e "\033[45;30m 紫底黑字 \033[0m"
        echo -e "\033[46;30m 天蓝底黑字 \033[0m"
        echo -e "\033[47;34m 白底蓝字 \033[0m"
    # 特效范围
        echo -e "\033[0m 无任何特效 \033[0m"
        echo -e "\033[1m 高亮度 \033[0m"
        echo -e "\033[4m 下划线 \033[0m"
        echo -e "\033[5m 闪烁 \033[0m"
```

- 课堂脚本：

```
#!/bin/bash
read -p  "`echo -e "\033[31;5m 请输入您的密码: \033[0m"`" pass
echo $pass

# 字体色范围：30-37
echo -e "\033[30m 黑色字 \033[0m"
```

```
echo -e "\033[31m 红色字 \033[0m"
echo -e "\033[32m 绿色字 \033[0m"
echo -e "\033[33m 黄色字 \033[0m"
echo -e "\033[34m 蓝色字 \033[0m"
echo -e "\033[35m 紫色字 \033[0m"
echo -e "\033[36m 天蓝字 \033[0m"
echo -e "\033[37m 白色字 \033[0m"

# 字背景颜色范围：40-47
echo -e "\033[40;37m 黑底白字 \033[0m"
echo -e "\033[41;30m 红底黑字 \033[0m"
echo -e "\033[42;34m 绿底蓝字 \033[0m"
echo -e "\033[43;34m 黄底蓝字 \033[0m"
echo -e "\033[44;30m 蓝底黑字 \033[0m"
echo -e "\033[45;30m 紫底黑字 \033[0m"
echo -e "\033[46;30m 天蓝底黑字 \033[0m"
echo -e "\033[47;34m 白底蓝字 \033[0m"

# 特效范围
echo -e "\033[0m 无任何特效 \033[0m"
echo -e "\033[1m 高亮度 \033[0m"
echo -e "\033[4m 下划线 \033[0m"
echo -e "\033[5m 闪烁 \033[0m"
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第五章：处理海量数据的grep、cut、awk、sed 命令

简介：分四节课

grep、cut、awk、sed 常常应用在查找日志、数据、输出结果等等，并对我们想要的数据进行提取，
通常grep，sed命令是对行进行提取，cut跟awk是对列进行提取

5.1、处理海量数据之grep命令

- grep应用场景：通常对数据进行 行的提取
- 语法：grep [选项]...[内容]...[file]

```
-v      #对内容进行取反提取

-n      #对提取的内容显示行号

-w      #精确匹配

-i      #忽略大小写

^       #匹配开头行首

-E      #正则匹配
```

5.2、处理海量数据之cut命令

- cut应用场景：通常对数据进行列的提取
- 语法：cut [选项]...[file]

```
-d  #指定分割符
-f  #指定截取区域
-c  #以字符为单位进行分割

注意：不加-d选项，默认为制表符，不是空格
/bin/bash       #代表可以登录的用户
/sbin/nologin   #代表不可以登录的用户

-d与-f
eg:
以':'为分隔符，截取出/etc/passwd的第一列跟第三列
cut -d ':' -f 1,3 /etc/passwd

eg:
以':'为分隔符，截取出/etc/passwd的第一列到第三列
cut -d ':' -f 1-3 /etc/passwd

eg:
以':'为分隔符，截取出/etc/passwd的第二列到最后一列
cut -d ':' -f 2- /etc/passwd

-c
eg:
截取/etc/passwd文件从第二个字符到第九个字符
cut -c 2-9 /etc/passwd
```

```
eg:比如领导想叫你截取linux上面所有可登陆普通用户
grep '/bin/bash' /etc/passwd | cut -d ':' -f 1 | grep -v root
```

5.3、处理海量数据之awk命令

```
awk的简介：其实一门编程语言，支持条件判断，数组，循环等功能，与grep，sed被称为linux三剑客
之所以叫AWK是因为取其三位创始人 Alfred Aho，Peter Weinberger，和 Brian Kernighan 的Family Name的首字符
```

- awk的应用场景：通常对数据进行列的提取
- 语法：

```
awk '条件 {执行动作}'文件名

awk '条件1 {执行动作} 条件2 {执行动作} ...' 文件名

或awk [选项] '条件1 {执行动作} 条件2 {执行动作} ...' 文件名
```

- 特殊要点与举例说明:

```
printf  #格式化输出，不会自动换行。
（%ns：字符串型，n代表有多少个字符； %ni：整型，n代表输出几个数字；%.nf：浮点型，n代表的是小数点后有多少个小数）

print   #打印出内容，默认会自动换行

\t      #制表符
\n      #换行符
eg：printf '%s\t%s\t%s\t%s\t%s\t%s\n' 1 2 3 4 5 6

eg：df -h |grep /dev/vda1 | awk '{printf "/dev/vda1的使用率是："} {print $5}'

小数：echo "scale=2; 0.13 + 0.1" | bc | awk '{printf "%.2f\n", $0}'

$1      #代表第一列
$2      #代表第二列
$0      #代表一整行
eg：  df -h | grep /dev/vda1 | awk '{print $5}'

-F      #指定分割符
eg：cat /etc/passwd | awk -F":" '{print $1}'


BEGIN   #在读取所有行内容前就开始执行，常常被用于修改内置变量的值
FS      #BEGIN时定义分割符
eg：cat /etc/passwd | awk 'BEGIN {FS=":"} {print $1}'

END     #结束的时候 执行
NR      #行号
eg：df -h | awk 'NR==2 {print $5}'
awk '(NR>=20 && NR<=30) {print $1}' /etc/passwd
```

5.4、处理海量数据之sed命令

- sed的应用场景：主要对数据进行处理（选取，新增，替换，删除，搜索）

- sed语法：sed [选项][动作] 文件名

```
常见的选项与参数：
-n   #把匹配到的行输出打印到屏幕
p    #以行为单位进行查询，通常与-n一起使用
eg：df -h | sed -n '2p'

d    #删除
eg: sed '2d' df.txt

a    #在行的下面插入新的内容
eg: sed '2a 1234567890' df.txt

i    #在行的上面插入新的内容
eg: sed '2i 1234567890' df.txt

c    #替换
eg: sed '2c 1234567890' df.txt

s/要被取代的内容/新的字符串/g   #指定内容进行替换
eg: sed 's/0%/100%/g' df.txt

-i   #对源文件进行修改(高危操作，慎用，用之前需要备份源文件)
搜索：在文件中搜索内容
eg：cat -n df.txt  | sed -n '/100%/p'

-e   #表示可以执行多条动作
eg：cat -n df.txt  | sed -n -e 's/100%/100%-----100%/g' -e '/100%-----100%/p'
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第六章：神奇的循环控制语句if 、for、case、while

简介：分为四节课，每一节课一个循环

6.1、if 循环控制（单分支与多分支）

- 单个判断（单分支循环）：

```
if [ 条件判断 ];
    then
    执行动作
fi

if [ 条件判断 ];
    then
    执行动作
    else
    执行动作
fi
```

- 多个判断（多分支循环）：

```
if [条件判断];
    then
    执行动作
    elif [条件判断];
    then
    执行动作
    elif [条件判断];
    then
    执行动作
fi
```

- 课堂脚本1：

```
#单分支循环
#!/bin/bash
# 判断文件是否存在

if [ -e 1.txt ];
    then
    echo '存在'
fi
```

- 课堂脚本2：

```
#单分支循环
#!/bin/bash
# 判断文件是否存在

if [ -e /home/$1 ];
        then
        echo '存在'
else
        echo '不存在'
fi
```

- 课堂脚本3：

```
#多分支循环：
#!/bin/bash
# 判断输入的数字是否大于10

echo '请输入一个数字：'
read number

if [ $等于number -eq 10 ];
    then
    echo '等于10'
elif [ $number -lt 10 ]
    then
    echo '小于10'
elif [ $number -gt 10 ]
    then
    echo '大于10'
fi
```

6.2、for循环控制

```
（1）for 变量名 in 值1 值2 值3
        do
        执行动作
        done
```

```
（2）for 变量名 in `命令`
        do
        执行动作
        done
```

```
（3）for ((条件))
        do
        执行动作
        done
```

- 课堂脚本1：

```
#!/bin/bash
# 依次打印1-10这10个数字

for  i  in 1 2 3 4 5 6 7 8 9 10
do
echo $i
sleep 2
done
```

- 课堂脚本2：

```
#!/bin/bash
# 依次打印1-10这10个数字

for  i  in `seq 1 10`
do
echo $i
sleep 2
done
```

- 课堂脚本3：
- a.txt:

```
www.baidu.com
www.taobao.com
www.taobaooooo.com
www.qq.com
```

```
#!/bin/bash
# ping 网站是否通

for i in $(cat a.txt)
do
ping -c 2 $i
echo -e "\n"
done
```

- 课堂脚本4：

```
#!/bin/bash
# 依次打印1-10这10个数字

for  (( i=1;i<11;i++ ))
do
echo $i
sleep 2
done
```

6.3、case循环控制

应用场景：case循环常使用于多重分支，与if不同的是，if可以判断多个条件，case一次只能判断一种条件

- 语法结构：

```
case 变量 in
```

```
值1 )
执行动作1
;;

值2 )
执行动作2
;;

值3 )
执行动作3
;;

....
esac
```

- 课堂脚本：

```
#!/bin/bash
# 程序选择

echo '请输入你要查询天气的城市：'
read city
case $city in
'广州')
echo '广州多云，温度：23~25摄氏度'
;;
'上海')
echo '上海下雨，温度：12~17摄氏度'
;;
'北京')
echo '北京下雪，温度：1~14摄氏度'
;;
'深圳')
echo '深圳多云，温度：22~26摄氏度'
;;
* )
echo '输入有误请重新输入'
;;

esac
```

6.4、while 循环

- 应用场景：

```
while循环是条件循环也是不定循环，只要条件判断式成立，循环就会一直进行着。直到判断式不成立 或者 选择跳出循环才会结束
```

- 语法结构：

```
while [ 条件判断式 ]
do
执行动作
done
```

- 课堂脚本：

```
#!/bin/bash
# 计算你从0加到输入的数字的总和一共是多少？

i=0
sum=0
while [ $i -lt $1 ]
do
sum=$(($sum+$i))
i=$(($i+1))
done
echo 'the sum is:'$sum
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第七章：shell的几个实战脚本例子

简介：分为5节课来讲，每一节课讲一个脚本，实现某个功能。

```
7.1、如何让shell实现 可选择性执行 的功能

7.2、巡检内存使用率

7.3、批量创建用户
      应用场景：公司想要做测试，需要10000个用户

7.4、数据库里查询学生成绩

7.5、如何实现高效率登录别的机器
      应用场景：假如公司有50台机器。每台机器对应ip不一样。
```

```
# 如何登录mysql数据库

# 如何写sql对数据进行操作

# 登录数据库（交互界面）/usr/local/mysql/bin/mysql -uroot -p

# 登录数据库（非交互界面）/usr/local/mysql/bin/mysql -uroot -p -e "sql"

# 展示所有的库：show database;

# 选择库：use student;
```

```
# 展示所有的表：show tables；

# 查询表的内容：select * from student.user where name='老王';
```

7.1、如何让shell实现 可选择性执行 的功能

- 课堂脚本：

```
#!/bin/bash
# 功能选项的选择

while [ 1 ]
do
cat <<EOF
**********************************************************************
* 1、算出你输入的目录下一共有多少文件 eg:/data                          *
* 2、计算从0加到你输入的数字为止eg：0+1+2+3+...+你输入的数字          *
* 3、批量创建用户                                                     *
* 4、测试用户名与密码是否匹配                                         *
* 5、测试ip通不通                                                     *
* 6、巡检内存使用率                                                   *
* 7、数据库里查询学生成绩                                             *
* q、退出                                                             *
**********************************************************************
EOF
echo '输入你想要的功能：'
read key
case $key in
1 )
clear
sh 1.sh
;;

2 )
clear
sh 2.sh
;;

q )
clear
echo '----感谢使用---程序退出----'
break
;;
esac
done
```

7.2、巡检内存使用率

- 课堂脚本：

```
#!/bin/bash
# 内存使用率
```

```
mem_total=`free -m | sed -n '2p' | awk '{print $2}'`

mem_used=`free -m | sed -n '2p' | awk '{print $3}'`

mem_free=`free -m | sed -n '2p' | awk '{print $4}'`

Percent_mem_used=`echo "scale=2; $mem_used / $mem_total * 100"  | bc`

Percent_mem_free=`echo "scale=2; $mem_free / $mem_total * 100"  | bc`

now_time=`date +"%Y-%m-%d %H:%M:%S 星期%w"`

echo -e "\n"

echo -e "$内存的使用率是：now_time\n内存的使用率是：$Percent_mem_used%"

echo -e "内存还剩：$Percent_mem_free%未使用"

# 检查负载是否有压力

if [ $告警：内存使用率已超过负载能力，目前使用率达到：mem_used -gt 1 ]
        then
        echo -e "\033[31;5m告警：\033[0m"
        echo -e "\033[31;5m内存使用率已超过负载能力，目前使用率达到：$Percent_mem_used%\033[0m"
else
        echo '目前内存负载正常'
fi

echo -e "\n"
```

7.3、批量创建用户

- 课堂脚本：

```
#!/bin/bash
# 批量创建用户

read -p '请输入创建的用户名称：' name
read -p '请输入创建用户的数量：' number

for (( i=1;i<=$number;i++ ))
do
# 需要查看系统是否存在用户
cat /etc/passwd | grep "${name}$i"  1>/dev/null
exist=`echo $?`
if [ $exist -eq 1 ]
    then
# 创建用户
echo -e "\n"
useradd ${name}$i 2>/dev/null && echo "创建用户${name}$i成功!"

# 需要生成随机密码（MD5值），MD5其实是一个算法来的，可以用来加密密码等
```

```
password=`head -2 /dev/urandom | md5sum | cut -c 1-8`

# 给新用户设置密码并把用户名跟密码放在文本中
echo $password | passwd --stdin ${name}$i 1>/dev/null  &&  echo -e "用户名：${name}$i\t 密
码：$password" >>/home/shell/newuser_password.txt
echo -e "\n"
else
    echo -e "\n"
    echo "${name}$i已经存在了，无需再创建！"
    echo -e "\n"

fi

done
```

7.4、数据库里查询学生成绩

- 课堂脚本：

```
# 如何登录mysql数据库

# 如何写sql对数据进行操作

# 登录数据库（交互界面）/usr/local/mysql/bin/mysql -uroot -p

# 登录数据库（非交互界面）/usr/local/mysql/bin/mysql -uroot -p -e "sql"

# 展示所有的库：show database;

# 选择库：use student;

# 展示所有的表：show tables；

# 查询表的内容：select * from student.user where name='老王';

# !/bin/bash

# 数据库查询

read -p '请输入你要查询的学生姓名：' sname
read -s -p  '请输入数据库用户：' user
echo -e "\n"
sql="select * from student.user where name='${sname}';"
mysql -u${user} -p -e "${sql}"
exit
```

7.5、如何实现高效率登录别的机器

- 课堂脚本：

```
#!/bin/bash
# 登录脚本

RegionIp=`cat /home/shell/ip.txt | grep $1 | awk -F "|" '{print $2}'`
ssh ${RegionIp}
```