优极限

“极限教育，挑战极限”

www.yjxxt.com

极限教育，挑战极限。优极限是一个让 95%的学生年薪过 18 万的岗前培训公司，让我们的学员具备优秀的互联网技术和职业素养，勇攀高薪，挑战极限。公司位于上海浦东，拥有两大校区，共万余平。累计培训学员超 3 万名。我们的训练营就业平均月薪 19000，最高年薪 50 万。

核心理念：让学员学会学习，拥有解决问题的能力，拿到高薪职场的钥匙。

项目驱动式团队协作、一对一服务、前瞻性思维、教练式培养模型-培养你成为就业明星。首创的老学员项目联盟给学员充分的项目、技术支撑，利用优极限平台这根杠杆，不断挑战极限，勇攀高薪，开挂人生。

扫码关注优极限微信公众号：

（获取最新技术相关资讯及更多源码笔记）

Shell编程

1. Shell编程概述

1.1. Shell名词解释

- Kernel
  - Linux内核主要是为了和硬件打交道
- Shell
  - 命令解释器(command interpreter)
  - Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。
  - Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。
- shell两大主流：
  - sh：
    - Bourne shell（sh），Solaris,hpux默认shell
    - Bourne again shell（bash），Linux系统默认shell
  - csh
    - C shell(csh)
    - tc shell(tcsh)
- **#!** 声明
  - 告诉系统其后路径所指定的程序即是解释此脚本文件的 Shell 程序
  - ```
    #!/bin/bash
    echo "Hello World !"
    ```

1.2. Shell脚本的执行

- 输入脚本的绝对路径或相对路径
  - /root/helloworld.sh
  - ./helloworld.sh
  - 执行的必须是一个可执行文件
- bash或sh +脚本
    ```

    - sh helloworld.sh
    - 当脚本没有x权限时，root和文件所有者通过该方式可以正常执行
- 在脚本的路径前再加". " 或source
    - source helloworld.sh
- 区别
    - 第一种和第二种会新开一个bash，不同bash中的变量无法共享。
    - 第三种 是在同一个shell里面执行的
- export ：可以将当前进程的变量传递给子进程去使用
    - 将来配置profile的时候 所有的变量前必须加export

2. Shell基础入门

2.1. shell变量

- 定义变量时，变量名不加美元符号
    - 命名只能使用英文字母，数字和下划线，首个字符不能以数字开头。
    - 中间不能有空格，可以使用下划线（_）。
    - 不能使用标点符号。
    - 不能使用bash里的关键字（可用help命令查看保留关键字）
- 变量的类型
    - 局部变量
        - 局部变量在脚本或命令中定义，仅在当前shell实例中有效，其他shell启动的程序不能访问局部变量。
    - 环境变量
        - 所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。
    - Shell变量
        - shell变量是由shell程序设置的特殊变量。shell变量中有一部分是环境变量，有一部分是局部变量
-
```
# 变量的声明
name="zhangsan"
for file in `ls /etc`
或
for file in $(ls /etc)

# 变量的调用
echo $name
echo ${name}

for skill in Ada Coffe Action Java; do
    echo "I am good at ${skill}Script"
done

# 只读变量  /bin/sh: NAME: This variable is read only.
url="https://www.google.com"
readonly url
url="https://www.runoob.com"

# 删除变量
unset name
```

2.2. Shell的字符串

- 字符串是shell编程中最常用最有用的数据类型，字符串可以用单引号，也可以用双引号，也可以不用引号。
- 单引号
  - 单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；
  - 单引号字串中不能出现单独一个的单引号，但可成对出现，作为字符串拼接使用。
- 双引号
  - 双引号里可以有变量
  - 双引号里可以出现转义字符
  - ```
    # 声明字符串
    str1="hello world 1"
    str2='hello world 2'

    # 字符串拼接--双引号
    name='sunwukong'
    name1="hello, "$name" !"
    name2="hello, ${name} !"

    # 字符串拼接--单引号
    passwd='123456'
    passwd1='hello, '$passwd' !'
    passwd2='hello, ${passwd} !'
    echo $passwd2  # hello, ${passwd} !

    # 字符串的长度
    email="123456@qq.com"
    echo ${#email}
    echo ${email:1:4}
    ```

2.3. Shell数组

- bash支持一维数组（不支持多维数组），并且没有限定数组的大小。
- 数组元素的下标由 0 开始编号。获取数组中的元素要利用下标，下标可以是整数或算术表达式，其值应大于或等于 0。
- ```
  # 定义数组 括号来表示数组，数组元素用"空格"符号分割开
  数组名=(值1 值2 ... 值n)
  favs=("足球" "蓝球" "乒乓球球" "保龄球")

  # 读取数组 ${数组名[下标]}
  fav=${favs[1]}

  # 使用 @ 符号可以获取数组中的所有元素
  echo ${favs[@]}

  # 获取数组的长度
  length1=${#favs[@]}
  length2=${#favs[*]}
  ```

2.4. Shell的注释

- 以 # 开头的行就是注释，会被解释器忽略。
- 通过每一行加一个 # 号设置多行注释
-
```
#--------------------------------------------
# 这是一个注释
# author：
# site：
#--------------------------------------------
##### 服务器配置-start #####
#
#
#
#
#
##### 服务器配置-end   #####


# 特殊的多行注释

:<<EOF
注释内容...
注释内容...
注释内容...
EOF

:<<!
注释内容...
注释内容...
注释内容...
!
```

2.5. Shell参数传递

- 执行 Shell 脚本时，向脚本传递参数，脚本内获取参数的格式为：**$n**。**n** 代表一个数字
-

参数处理	参数说明
$#	传递到脚本的参数个数
$*	以一个单字符串显示所有向脚本传递的参数。
$$	脚本运行的当前进程ID号
$!	后台运行的最后一个进程的ID号
$?	显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。
$0	执行的文件名

- ```
  #!/bin/bash

  echo "Shell 传递参数实例！";
  echo "执行的文件名：$0";
  echo "第一个参数为：$1";
  echo "第二个参数为：$2";
  echo "第三个参数为：$3";

  # ./hello.sh 11 22 33 44
  ```

3. Shell高级进阶

3.1. Shell运算符

- 运算符的分类
  - 算数运算符

运算符	说明	举例
+	加法	`expr $a + $b` 结果为 30。
-	减法	`expr $a - $b` 结果为 -10。
*	乘法	`expr $a \* $b` 结果为 200。
/	除法	`expr $b / $a` 结果为 2。
%	取余	`expr $b % $a` 结果为 0。
=	赋值	a=$b 将把变量 b 的值赋给 a。
==	相等。用于比较两个数字，相同则返回 true。	[$a == $b] 返回 false。
!=	不相等。用于比较两个数字，不相同则返回 true。	[$a != $b] 返回 true。

  - 关系运算符

运算符	说明	举例
-eq	检测两个数是否相等，相等返回 true。	[$a -eq $b] 返回 false。
-ne	检测两个数是否不相等，不相等返回 true。	[$a -ne $b] 返回 true。
-gt	检测左边的数是否大于右边的，如果是，则返回 true。	[$a -gt $b] 返回 false。
-lt	检测左边的数是否小于右边的，如果是，则返回 true。	[$a -lt $b] 返回 true。
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true。	[$a -ge $b] 返回 false。
-le	检测左边的数是否小于等于右边的，如果是，则返回 true。	[$a -le $b] 返回 true。

  - 布尔运算符

运算符	说明	举例
!	非运算，表达式为 true 则返回 false，否则返回 true。	[! false] 返回 true。
-o	或运算，有一个表达式为 true 则返回 true。	[$a -lt 20 -o $b -gt 100] 返回 true。
-a	与运算，两个表达式都为 true 才返回 true。	[$a -lt 20 -a $b -gt 100] 返回 false。

  - 字符串运算符

运算符	说明	举例
&&	逻辑的 AND	[[$a -lt 100 && $b -gt 100]] 返回 false
||	逻辑的 OR	[[$a -lt 100 || $b -gt 100]] 返回 true

  - 文件测试运算符
  ```

运算符	说明	举例
=	检测两个字符串是否相等，相等返回 true。	[$a = $b] 返回 false。
!=	检测两个字符串是否相等，不相等返回 true。	[$a != $b] 返回 true。
-z	检测字符串长度是否为0，为0返回 true。	[-z $a] 返回 false。
-n	检测字符串长度是否不为 0，不为 0 返回 true。	[-n "$a"] 返回 true。
$	检测字符串是否为空，不为空返回 true。	[$a] 返回 true。

3.1.1. 算数运算符

- expr 是一款表达式计算工具，使用它能完成表达式的求值操作。
- ```
  #!/bin/bash
  a=10
  b=20

  val=`expr $a + $b`
  echo "a + b : $val"

  val=`expr $a - $b`
  echo "a - b : $val"

  val=`expr $a \* $b`
  echo "a * b : $val"

  val=`expr $b / $a`
  echo "b / a : $val"

  val=`expr $b % $a`
  echo "b % a : $val"

  if [ $a == $b ]
  then
     echo "a 等于 b"
  fi

  if [ $a != $b ]
  then
     echo "a 不等于 b"
  fi
  ```

3.1.2. 关系运算符

- 关系运算符只支持数字，不支持字符串，除非字符串的值是数字。
- ```
  #!/bin/bash

  a=10
  b=20

  if [ $a -eq $b ]
  then
     echo "$a -eq $b : a 等于 b"
  else
     echo "$a -eq $b: a 不等于 b"
  fi
  ```

```
if [ $a -ne $b ]
then
   echo "$a -ne $b: a 不等于 b"
else
   echo "$a -ne $b : a 等于 b"
fi

if [ $a -gt $b ]
then
   echo "$a -gt $b: a 大于 b"
else
   echo "$a -gt $b: a 不大于 b"
fi

if [ $a -lt $b ]
then
   echo "$a -lt $b: a 小于 b"
else
   echo "$a -lt $b: a 不小于 b"
fi

if [ $a -ge $b ]
then
   echo "$a -ge $b: a 大于或等于 b"
else
   echo "$a -ge $b: a 小于 b"
fi

if [ $a -le $b ]
then
   echo "$a -le $b: a 小于或等于 b"
else
   echo "$a -le $b: a 大于 b"
fi
```

3.1.3. 布尔运算符

- ```
  #!/bin/bash

  a=10
  b=20

  if [ $a != $b ]
  then
     echo "$a != $b : a 不等于 b"
  else
     echo "$a == $b: a 等于 b"
  fi

  if [ $a -lt 100 -a $b -gt 15 ]
  then
     echo "$a 小于 100 且 $b 大于 15 : 返回 true"
  else
     echo "$a 小于 100 且 $b 大于 15 : 返回 false"
  fi

  if [ $a -lt 100 -o $b -gt 100 ]
  ```
  ```

```
then
   echo "$a 小于 100 或 $b 大于 100 : 返回 true"
else
   echo "$a 小于 100 或 $b 大于 100 : 返回 false"
fi

if [ $a -lt 5 -o $b -gt 100 ]
then
   echo "$a 小于 5 或 $b 大于 100 : 返回 true"
else
   echo "$a 小于 5 或 $b 大于 100 : 返回 false"
fi
```

3.1.4. 逻辑运算符

- ```
  #!/bin/bash

  a=10
  b=20

  if [[ $a -lt 100 && $b -gt 100 ]]
  then
     echo "返回 true"
  else
     echo "返回 false"
  fi

  if [[ $a -lt 100 || $b -gt 100 ]]
  then
     echo "返回 true"
  else
     echo "返回 false"
  fi
  ```

3.1.5. 字符串运算符

- ```
  #!/bin/bash

  a="abc"
  b="efg"

  if [ $a = $b ]
  then
     echo "$a = $b : a 等于 b"
  else
     echo "$a = $b: a 不等于 b"
  fi
  if [ $a != $b ]
  then
     echo "$a != $b : a 不等于 b"
  else
     echo "$a != $b: a 等于 b"
  fi
  if [ -z $a ]
  then
     echo "-z $a : 字符串长度为 0"
  ```

```
else
   echo "-z $a : 字符串长度不为 0"
fi
if [ -n "$a" ]
then
   echo "-n $a : 字符串长度不为 0"
else
   echo "-n $a : 字符串长度为 0"
fi
if [ $a ]
then
   echo "$a : 字符串不为空"
else
   echo "$a : 字符串为空"
fi
```

3.1.6. 文件测试运算符

- ```
  #!/bin/bash

  file="/var/node/test.sh"
  if [ -r $file ]
  then
     echo "文件可读"
  else
     echo "文件不可读"
  fi
  if [ -w $file ]
  then
     echo "文件可写"
  else
     echo "文件不可写"
  fi
  if [ -x $file ]
  then
     echo "文件可执行"
  else
     echo "文件不可执行"
  fi
  if [ -f $file ]
  then
     echo "文件为普通文件"
  else
     echo "文件为特殊文件"
  fi
  if [ -d $file ]
  then
     echo "文件是个目录"
  else
     echo "文件不是个目录"
  fi
  if [ -s $file ]
  then
     echo "文件不为空"
  else
     echo "文件为空"
  fi
  ```
  ```

```
if [ -e $file ]
then
   echo "文件存在"
else
   echo "文件不存在"
fi
```

3.2. echo打印数据

- Shell 的 echo 指令用于字符串的输出。
- ```
  ## 显示普通字符串
  echo "Hello World"
  ## 显示转义字符
  echo "\"Hello World\""
  ## 显示变量
  name="zhangsan"
  echo "$name Hello World"
  ## 显示换行
  echo -e "OK! \n"
  echo "Hello World"
  ## 显示不换行
  echo -e "OK! \c"
  echo "Hello World"
  ## 显示结果定向至文件
  echo "Hello World" > myfile
  ## 原样输出字符串
  echo '$name\"'
  ## 显示命令执行结果
  echo `date`
  ```

3.3. test 命令

- Shell中的 test 命令用于检查某个条件是否成立，它可以进行数值、字符和文件三个方面的测试。
- 数字
参数	说明
-eq	等于则为真
-ne	不等于则为真
-gt	大于则为真
-ge	大于等于则为真
-lt	小于则为真
-le	小于等于则为真
- 字符串
参数	说明
=	等于则为真
!=	不相等则为真
-z 字符串	字符串的长度为零则为真
-n 字符串	字符串的长度不为零则为真
- 文件测试
  ```

参数	说明
-e 文件名	如果文件存在则为真
-r 文件名	如果文件存在且可读则为真
-w 文件名	如果文件存在且可写则为真
-x 文件名	如果文件存在且可执行则为真
-s 文件名	如果文件存在且至少有一个字符则为真
-d 文件名	如果文件存在且为目录则为真
-f 文件名	如果文件存在且为普通文件则为真
-c 文件名	如果文件存在且为字符型特殊文件则为真
-b 文件名	如果文件存在且为块特殊文件则为真

```
# 比较
num1=100
num2=100
if test $[num1] -eq $[num2]
then
    echo '两个数相等！'
else
    echo '两个数不相等！'
fi
```

3.4. Shell流程控制

3.4.1. if

```
if condition1
then
    command1
elif condition2
then
    command2
else
    commandN
fi
```

```
a=10
b=20
if [ $a == $b ]
then
   echo "a 等于 b"
elif [ $a -gt $b ]
then
   echo "a 大于 b"
elif [ $a -lt $b ]
then
   echo "a 小于 b"
else
   echo "没有符合的条件"
fi
```

3.4.2. case

- Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。
- ```
  case 值 in
  模式1)
      command1
      command2
      ...
      commandN
      ;;
  模式2）
      command1
      command2
      ...
      commandN
      ;;
  esac


  echo '输入 1 到 4 之间的数字:'
  echo '你输入的数字为:'
  read num
  case $num in
      1)  echo '你选择了 1'
      ;;
      2)  echo '你选择了 2'
      ;;
      3)  echo '你选择了 3'
      ;;
      4)  echo '你选择了 4'
      ;;
      *)  echo '你没有输入 1 到 4 之间的数字'
      ;;
  esac
  ```

3.4.3. for

- 当变量值在列表里，for循环即执行一次所有命令，使用变量名获取列表中的当前取值。
- 命令可为任何有效的shell命令和语句。in列表可以包含替换、字符串和文件名。
- in列表是可选的，如果不用它，for循环使用命令行的位置参数。
- ```
  for var in item1 item2 ... itemN
  do
      command1
      command2
      ...
      commandN
  done
  ```

- ```
  for loop in 1 2 3 4 5
  do
      echo "The value is: $loop"
  done

  for str in 'This is a string','hello moto'
  do
      echo $str
  done
  ```

3.4.4. while循环

- while循环用于不断执行一系列命令，也用于从输入文件中读取数据；命令通常为测试条件。
- ```
  while condition
  do
      command
  done
  ```
- ```
  #  Bash let 命令，它用于执行一个或多个表达式，变量计算中不需要加上 $ 来表示变量
  #!/bin/bash
  int=1
  while(( $int<=5 ))
  do
      echo $int
      let "int++"
  done

  # 无限循环

  while true
  do
      command
  done
  ```

3.4.5. break

- break命令允许跳出所有循环（终止执行后面的所有循环）。
- ```
  #!/bin/bash
  while :
  do
      echo -n "输入 1 到 5 之间的数字:"
      read aNum
      case $aNum in
          1|2|3|4|5) echo "你输入的数字为 $aNum!"
          ;;
          *) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"
              break
          ;;
      esac
  done
  ```

3.4.6. continue

- continue命令不会跳出所有循环，仅仅跳出当前循环。
- ```
  #!/bin/bash
  while :
  do
      echo -n "输入 1 到 5 之间的数字: "
      read aNum
      case $aNum in
          1|2|3|4|5) echo "你输入的数字为 $aNum!"
          ;;
          *) echo "你输入的数字不是 1 到 5 之间的!"
              continue
              echo "游戏结束"
          ;;
      esac
  done
  ```

3.5. Shell函数

- linux shell 可以用户定义函数，然后在shell脚本中可以随便调用。
- 可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。
- 参数返回，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值n(0-255
- ```
  #!/bin/bash

  ## 第一个函数------------------------------
  demoFun(){
      echo "这是我的第一个 shell 函数!"
  }
  echo "-----函数开始执行-----"
  demoFun
  echo "-----函数执行完毕-----"

  ## 函数返回值------------------------------
  funWithReturn(){
      echo "这个函数会对输入的两个数字进行相加运算..."
      echo "输入第一个数字: "
      read aNum
      echo "输入第二个数字: "
      read anotherNum
      echo "两个数字分别为 $aNum 和 $anotherNum !"
      return $(($aNum+$anotherNum))
  }
  funWithReturn
  # 函数返回值在调用该函数后通过 $? 来获得。
  echo "输入的两个数字之和为 $? !"

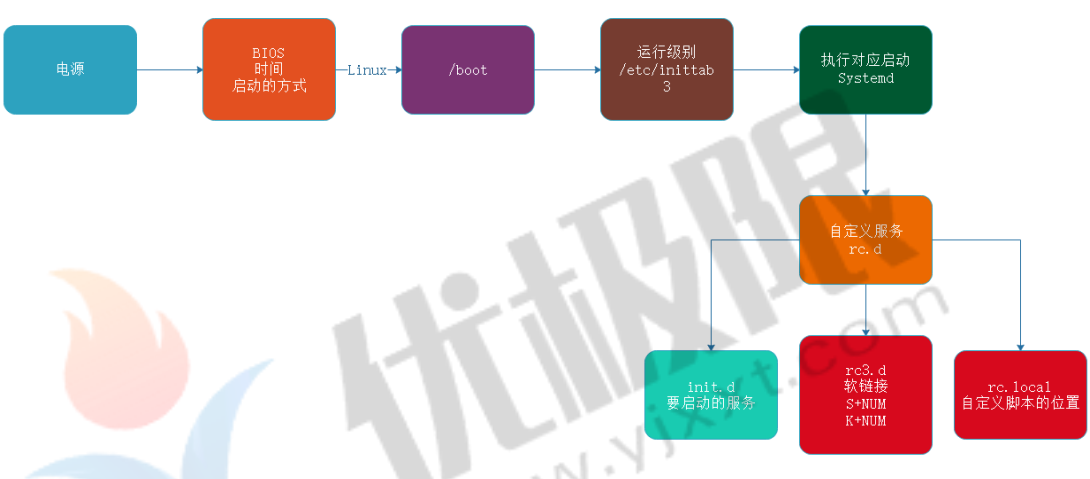
  ## 函数参数------------------------------
  funWithParam(){
      echo "第一个参数为 $1 !"
      echo "第二个参数为 $2 !"
      echo "第十个参数为 $10 !"
      echo "第十个参数为 ${10} !"
  ```

```
    echo "第十一个参数为 ${11} !"
    echo "参数总数有 $# 个!"
    echo "作为一个字符串输出所有参数 $* !"
}
funWithParam 1 2 3 4 5 6 7 8 9
```

4. 系统任务设置

4.1. 系统启动流程

- 启动计算机的硬件(BIOS)
  - 读取时间
  - 选择对应的启动模式(USB HDD EFI)
- 如果是Linux系统，回去找/boot目录.引导这个系统启动
- 计算机系统开始启动,读取初始化配置文件
  - vim /etc/inittab
  - 启动时控制着计算机的运行级别 runlevel
0	**halt (关机)**
1	Single user mode(单用户模式)
2	Multiuser, without NFS(多用户模式，但是无网络状态) FS-->FileSystem
3	Full multiuser mode(多用户完整版模式)
4	unused (保留模式)
5	X11(用户界面模式)
6	reboot(重启模式)
  - id:3:initdefault: 默认runlevel为3
  - 以runlevel=3开始启动对应的服务和组件
- 开始默认引导公共的组件或者服务
  - vim /etc/rc.d/rc.sysinit
- 开始加载对应runlevel的服务
  - vi /etc/rc3.d/
    - K:关机时需要关闭的服务
    - S:启动时需要开启的服务
    - 数字代表了开启或者关闭的顺序
    - 所有的文件都是软链接，链接的地址为 /etc/init.d
- 当启动完毕，所有的服务也被加载完成
-



4.2. 系统服务

- 我们可以使用chkconfig命令查看当前虚拟机的服务
- 通过查看可以得知不同的级别对应到每一个服务确定本次开机自动启动
- 开机结束后，我们需要使用service（Centos6）Systemctl(Centos7)命令控制服务的开启或者关闭

4.3. 开机自启动服务

- rc.local
  - 首先创建脚本存放的文件夹
    - mkdir -p /usr/local/scripts
  - 在文件夹中创建脚本文件
    - vim hello.sh
    - 给予执行权限
  - 去/etc/rc.d/rc.local文件中添加脚本的绝对路径
    - 给予rc.local执行权限
  - 修改系统时间
    - date -s '21-21-21'
  - 重启虚拟机
    - reboot
- chkconfig
  - 创建开机自启动脚本文件
  - vim schoolntpdate.sh
  - ```
    #!/bin/bash
    #chkconfig: 2345 88 99
    #description:auto_run

    # 开机自启动同步时间
    yum info ntp && ntpdate cn.ntp.org.cn
    ```
  - 给其设置执行权限
    - chmod u+x schoolntpdate.sh
  - 将脚本拷贝到 /etc/init.d目录下
    - cp schoolntpdate.sh /etc/init.d/
  - 添加到服务
    - chkconfig --add /etc/init.d/schoolntpdate.sh
    ```

- 修改系统时间
  - date -s '21-21-21'
- 重启服务器
  - reboot

4.4. 定时任务

- 在系统服务中心，crond负责周期任务
  - systemctl status crond.service
- 添加任务，编辑当前用户的任务列表
  - crontab -e
- 编辑任务
  - 星 星 星 星 星 command
    分 时 日 月 周 命令
    第1列表示分钟1～59 每分钟用*或者 */2表示
    第2列表示小时1～23（0表示0点）
    第3列表示日期1～31
    第4列表示月份1～12
    第5列标识号星期0～6（0表示星期天）
    第6列要运行的命令

    *：表示任意时间都，实际上就是“每”的意思。可以代表00-23小时或者00-12每月或者00-59分
    -：表示区间，是一个范围，00 17-19 * * * cmd，就是每天17,18,19点的整点执行命令
    ,：是分割时段，30 3,19,21 * * * cmd，就是每天凌晨3和晚上19,21点的半点时刻执行命令
    /n：表示分割，可以看成除法，*/5 * * * * cmd，每隔五分钟执行一次
  - ```
    30 21 * * * /usr/local/etc/rc.d/lighttpd restart
    上面的例子表示每晚的21:30重启apache。

    45 4 1,10,22 * * /usr/local/etc/rc.d/lighttpd restart
    上面的例子表示每月1、10、22日的4 : 45重启apache。

    10 1 * * 6,0 /usr/local/etc/rc.d/lighttpd restart
    上面的例子表示每周六、周日的1 : 10重启apache。

    0,30 18-23 * * * /usr/local/etc/rc.d/lighttpd restart
    上面的例子表示在每天18 : 00至23 : 00之间每隔30分钟重启apache。

    0 23 * * 6 /usr/local/etc/rc.d/lighttpd restart
    上面的例子表示每星期六的11 : 00 pm重启apache。

    * */2 * * * /usr/local/etc/rc.d/lighttpd restart
    每两小时重启apache

    * 23-7/1 * * * /usr/local/etc/rc.d/lighttpd restart
    晚上11点到早上7点之间，每隔一小时重启apache

    0 11 4 * mon-wed /usr/local/etc/rc.d/lighttpd restart
    每月的4号与每周一到周三的11点重启apache

    0 4 1 jan * /usr/local/etc/rc.d/lighttpd restart
    一月一号的4点重启apache
    ```
    ```

```
--（功能描述：显示年月日时分秒）
date "+%Y%m%d%H%M%S"
```

- 重启crontab，使配置生效
  - systemctl restart crond.service
- 通过crontab -l
  - 查看当前的定时任务
- 查看任务的历史
  - vim /var/spool/mail/root
- 清除任务
  - crontab -r

5. 虚拟机初始化脚本

```
#!/bin/bash
## -bash: ./lucky.sh: /bin/bash^M: bad interpreter: No such file or directory
## vim或者vi的命令模式下，输入命令  set fileformat=unix 即可解决换行问题
echo -e "\e[1;31m【----------------------------------------在opt和var创建lucky文件夹】\e[0m"
sleep 5
mkdir -p /opt/lucky
mkdir -p /var/lucky
mkdir -p /usr/local/script

echo -e "\e[1;31m【----------------------------------------禁用防火墙】\e[0m"
sleep 5
systemctl stop firewalld
systemctl disable firewalld
systemctl status firewalld

echo -e "\e[1;32m【----------------------------------------修改selinux】\e[0m"
sleep 5
sed -i '/^SELINUX=/c SELINUX=disabled' /etc/selinux/config

echo -e "\e[1;32m【----------------------------------------安装wget】\e[0m"
sleep 5
yum install wget -y

echo -e "\e[1;33m【----------------------------------------修改yum源】\e[0m"
sleep 5
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
yum clean all
yum makecache

echo -e "\e[1;33m【----------------------------------------安装常用软件】\e[0m"
yum install man man-pages ntp vim lrzsz zip unzip telnet perl net-tools -y

echo -e "\e[1;34m【----------------------------------------同步系统时间】\e[0m"
yum info ntp &&  ntpdate cn.ntp.org.cn

echo -e "\e[1;34m【----------------------------------------DNS域名配置】\e[0m"
sleep 5
```

```
echo "192.168.58.100 basenode" >> /etc/hosts
echo "192.168.58.161 bd1601" >> /etc/hosts
echo "192.168.58.162 bd1602" >> /etc/hosts
echo "192.168.58.163 bd1603" >> /etc/hosts
echo -e "\e[1;34m【----------------------------------------安装JDK】\e[0m"
sleep 5
rpm -ivh jdk-8u231-linux-x64.rpm
echo 'export JAVA_HOME=/usr/java/jdk1.8.0_231-amd64' >> /etc/profile
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /etc/profile
source /etc/profile
echo -e "\e[1;35m【----------------------------------------安装Tomcat】\e[0m"
sleep 5
tar -zxf apache-tomcat-8.5.47.tar.gz
mv apache-tomcat-8.5.47 /opt/lucky/
echo -e "\e[1;35m【----------------------------------------安装Mysql】\e[0m"
sleep 5
rpm -e --nodeps `rpm -qa | grep mariadb`

tar -xvf mysql-5.7.28-1.el7.x86_64.rpm-bundle.tar
rpm -ivh mysql-community-common-5.7.28-1.el7.x86_64.rpm
rpm -ivh mysql-community-libs-5.7.28-1.el7.x86_64.rpm
rpm -ivh mysql-community-client-5.7.28-1.el7.x86_64.rpm
rpm -ivh mysql-community-server-5.7.28-1.el7.x86_64.rpm

systemctl start mysqld
systemctl enable mysqld

temppasswd=`grep "A temporary password" /var/log/mysqld.log | awk '{ print
$NF}'`

mysql -uroot -p$temppasswd --connect-expired-password << EOF
set global validate_password_policy=low;
set global validate_password_length=6;
alter user root@localhost identified by '123456';

use mysql;
update user set  host='%' where user = 'root';
commit;
quit
EOF

systemctl restart mysqld

echo -e "\e[1;35m【----------------------------------------安装Nginx】\e[0m"
sleep 5
echo -e "\e[1;36m【----------------------------------------设置开机启动项】\e[0m"
sleep 5
touch /usr/local/script/auto_ntpdate.sh
echo '#!/bin/bash' >> /usr/local/script/auto_ntpdate.sh
echo 'yum info ntp &&  ntpdate cn.ntp.org.cn' >>
/usr/local/script/auto_ntpdate.sh
chmod u+x /usr/local/script/auto_ntpdate.sh

echo '/usr/local/script/auto_ntpdate.sh' >> /etc/rc.local
chmod u+x /etc/rc.local

echo -e "\e[1;36m【----------------------------------------删除文件】\e[0m"
sleep 5
```

```
rm -rf apache-tomcat-8.5.47.tar.gz
rm -rf jdk-8u231-linux-x64.rpm
rm -rf mysql*
rm -rf *.sh

echo -e "\e[1;36m【----------------------------------------关闭计算器，拍快照】
\e[0m"
sleep 5
shutdown -h now
```

6. 虚拟机相互免秘钥

```
##三台主机分别生成秘钥
【123】ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

##host验证
【123】vim /etc/ssh/ssh_config  在最后添加

  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null

##将秘钥分别拷贝给自己和别人
【123】ssh-copy-id -i ~/.ssh/id_rsa.pub root@bd1701
【123】ssh-copy-id -i ~/.ssh/id_rsa.pub root@bd1702
【123】ssh-copy-id -i ~/.ssh/id_rsa.pub root@bd1703

123456

##关闭主机拍摄快照
power off
```