

# linux实战篇（四）

作者：强哥

版本：QF1.0

版权：千锋java教研员

## 第一章：软件安装

### 1.1、rpm软件包管理器

#### 目标

- 通过 `rpm`命令 实现对软件 的安装、查询、卸载
- RPM 是Red-Hat Package Manager（RPM软件包管理器）的缩写
- 虽然 打上了 red-hat 的标记, 但是理念开放, 很多发行版都采用, 已经成为行业标准

#### 路径

- 第一步: rpm包 的查询命令
- 第二步: rpm包 的 卸载
- 第三步: rpm包 的 安装

#### 3 实现

#### 第一步: rpm包 的 查询命令

选项	英文	含义
<code>-q</code>	query	查询
<code>-a</code>	all	所有
<code>-i</code>	info	信息
<code>-l</code>	list	显示所有相关文件
<code>-f</code>	file	文件, 显示文件对应 <code>rpm</code> 包

- 查询已安装的rpm列表

```
1 rpm -qa | grep XXX
2 rpm -qa | less
```

- 查询软件包信息

```
1 rpm -qi 软件全包名
```

- 查看一个rpm包中的文件安装到哪里去了?

```
1 rpm -ql 软件全包名
```

- 查看指定文件归属于那个软件包

```
1 rpm -qf 文件的全路径
```

## 第二步: rpm包的 卸载

命令	英文	含义
<code>rpm -e 软件包名称</code>	erase 清除	卸载rpm软件包
<code>rpm -e --nodeps 软件包名称</code>	Don't check dependencies	卸载前 跳过 依赖检查

### 第三步: rpm包的 安装

命令	含义
<code>rpm -ivh rpm包的全路径</code>	安装 rpm 包

参数	英文	含义
<code>-i</code>	install	安装
<code>-v</code>	verbose	打印提示信息
<code>-h</code>	hase	显示安装进度

#### 小结

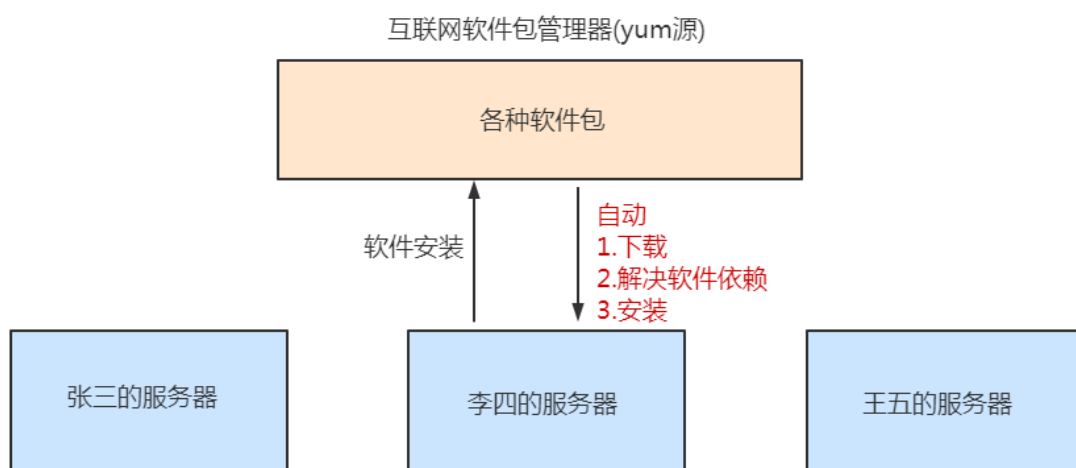
```
1 # 1 查询
2 rpm -qa | grep rpm包
3
4 # 2 卸载
5 rpm -e rpm全包名
6 rpm -e --nodeps rpm全包名
7
8 # 3 安装
9 rpm -ivh rpm包的全路径
10
```

## 1.2、yum安装与使用

### yum介绍

yum 是一个交互式的、基于 rpm 的包管理器，是一种联网安装软件的方式。它可以安装软件、卸载软件、查询软件、更新软件以及系统、自动解决rpm包依赖问题

## yum原理



## 更新yum源

### 备份旧的软件源配置文件

```
1 mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
```

- 1 防止新的软件源文件无法使用，导致系统无法安装软件，需要提前备份软件源配置文件，操作配置文件前备份是一个好的习惯。

### 下载软件源配置文件

```
1 wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
```

## yum软件源更新

```
1 yum clean all
2 yum makecache
```

## yum中指令

### 注意: 必须联网

命令	含义
<code>yum list   grep 需要的软件名</code>	yum list显示所有已经安装和可以安装的程序包
<code>yum -y install 需要的软件包</code>	下载安装
<code>yum -y remove 需要卸载的软件包</code>	卸载
<code>yum repolist</code>	列出设定yum源信息
<code>yum clean all</code>	清除yum缓存信息

### 1.3、安装httpd软件

- 安装httpd:

```
1 yum -y install httpd
```

- 启动 httpd 服务

```
1 systemctl start httpd
```

- 测试

```
1 http://192.168.80.100:80
```

注意：关闭防火墙

```
1 systemctl stop firewalld
```

## 1.4、安装JDK11

系统默认安装OpenJDK，首先需要删除OpenJDK

### 1、查看以前是不是安装了openjdk

如果不是root用户需要切换到root用户（su - root）

命令：`rpm -qa | grep java`

显示如下：（有则卸载，没有就不用），注意版本可能会有些不一样，以实际操作的为准。

```
1 tzdata-java-2013g-1.el6.noarch
2 java-1.7.0-openjdk-1.7.0.45-2.4.3.3.el6.x86_64
3 java-1.6.0-openjdk-1.6.0.0-1.66.1.13.0.el6.x86_64
```

### 2、卸载openjdk:

（其中参数“tzdata-java-2013j-1.el6.noarch”为上面查看中显示的结果，粘进来就行，如果你显示的不一样，请复制你查询到的结果）

```
rpm -e --nodeps java-1.6.0-openjdk-1.6.0.0-1.66.1.13.0.el6.i686
```

```
rpm -e --nodeps java-1.7.0-openjdk-1.7.0.45-2.4.3.3.el6.i686
```

```
rpm -e --nodeps tzdata-java-2013g-1.el6.noarch
```

```
1 -e          erase package(uninstall)      移除包(卸载)
2 --nodeps    do not verify package dependencies  不要验证包依赖关系
```

### 3、安装jdk

(1)、切换到root用户并进入usr目录：`cd /usr`

(2)、在usr目录下创建java文件夹：`mkdir java`

(3)、将tar -zxvf jdk-11.0.15\_linux-x64\_bin.tar.gz拷贝或上传到java目录下（也可以用工具）

(4)、进入/usr/java文件夹下: `cd /usr/java/`

(5)、修改权限, 参数“jdk-11.0.15\_linux-x64\_bin.tar”为你自己上传的jdk安装文件

`chmod 755 jdk-11.0.15_linux-x64_bin.tar`

(6)、解压: `tar -zxvf jdk-11.0.15_linux-x64_bin.tar.gz`

(7)、配置环境变量

```
1 | vi /etc/profile
```

添加内容:

`export` 主要用来设置环境变量

```
1 | export JAVA_HOME=/usr/java/jdk-11.0.15/
2 | export
  CLASSPATH=.:${JAVA_HOME}/jre/lib/rt.jar:${JAVA_HOME}/lib/dt.jar
  :${JAVA_HOME}/lib/tools.jar
3 | export PATH=$PATH:${JAVA_HOME}/bin
```

(8)、重新编译环境变量

```
1 | source /etc/profile
```

(9)、测试

```
1 | java -version
```

## 1.5、安装tomcat8

### 1. 下载安装包

<https://tomcat.apache.org/download-80.cgi>

Tomcat Connectors

Tomcat Native

Taglibs

Archives

Documentation

Tomcat 10.1 (alpha)

Tomcat 10.0

Tomcat 9.0

Tomcat 8.5

Tomcat Connectors

Tomcat Native

Wiki

Migration Guide

Presentations

Specifications

Problems?

Security Reports

Find help

FAQ

Mailing Lists

Bug Database

IRC

Get Involved

OpenPGP keys of Tomcat's Release Managers. We also provide [SHA-512](#) checksums for every release file. After you download the file, you should calculate a checksum for your download, and make sure it is the same as ours.

Mirrors

You are currently using <https://dlcdn.apache.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors:

8.5.79

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - [zip \(pgp, sha512\)](#)
  - [tar.gz \(pgp, sha512\)](#)
  - [32-bit Windows zip \(pgp, sha512\)](#)
  - [64-bit Windows zip \(pgp, sha512\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
  - [tar.gz \(pgp, sha512\)](#)

## 2.解压

等待解压完成即可

```
1 tar -zxvf apache-tomcat-8.5.53.tar.gz -C /user/tomcat
```

## 3.启动

进入tomcat的bin目录：`cd /usr/tomcat/apache-tomcat-8.5.53/bin/`


启动tomcat web服务器：`./startup.sh`

访问：`192.168.80.100:8080`



[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#) [Find Help](#)

## Apache Tomcat/8.5.53



If you're seeing this, you've successfully installed Tomcat. Congratulations!

**Recommended Reading:**

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

[Server Status](#)  
[Manager App](#)  
[Host Manager](#)

### Developer Quick Start

<a href="#">Tomcat Setup</a> <a href="#">First Web Application</a>	<a href="#">Realms &amp; AAA</a> <a href="#">JDBC DataSources</a>	<a href="#">Examples</a>	<a href="#">Servlet Specifications</a> <a href="#">Tomcat Versions</a>
-----------------------------------------------------------------------	----------------------------------------------------------------------	--------------------------	---------------------------------------------------------------------------

### Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 8.5 access to the manager application is split between different users.  
[Read more...](#)

[Release Notes](#)  
[Changelog](#)

### Documentation

[Tomcat 8.5 Documentation](#)  
[Tomcat 8.5 Configuration](#)  
[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

[Tomcat 8.5 Run Database](#)

### Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)  
Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)  
User support and discussion

[taglibs-user](#)  
User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)

## 4.停止tomcat

: `./shutdown.sh`

## 5.查看tomcat日志信息:

```
1 tail -200f /usr/tomcat/apache-tomcat-8.5.53/logs/catalina.out
```

200表示最后显示行数

也可以用组合命令，启动并查看日志：

```
1 ./startup.sh && tail -200f ../logs/catalina.out
```

## 1.6、安装mysql

在CentOS7安装MySQL8.0.X，有两种情况：

新安装的系统，从未安装过MySQL

对于第一种情况，我们需要把MySQL彻底卸载删除干净，重新安装；  
对于第二种情况，卸载系统自带的mariadb，再安装MySQL就行。

## 1.彻底卸载删除MYSQL

卸载

```
1 | yum -y remove mysql*
```

查找残余的组件

```
1 | rpm -qa | grep -i mysql
```

如果有残余组件就删掉

```
1 | yum remove mysql-community*
```

查找残余目录

```
1 | whereis mysql
```

如果有残余的目录就删掉

```
1 | rm -rf XXX
```

继续查找是否有配置文件之类的

```
1 | find / -name mysql
```

```
[root@localhost ~]# find / -name mysql
/etc/selinux/targeted/active/modules/100/mysql
/etc/selinux/targeted/tmp/modules/100/mysql
/var/lib/mysql
/var/lib/mysql/mysql
/mysql
[root@localhost ~]#
```

把找到的都用 `rm -rf xxx` 删除

```
[root@localhost ~]# rm -rf /etc/selinux/targeted/active/modules/100/mysql/
[root@localhost ~]# rm -rf /etc/selinux/targeted/tmp/modules/100/mysql/
[root@localhost ~]# rm -rf /var/lib/mysql/
[root@localhost ~]# find / -name mysql
/mysql
[root@localhost ~]# rm -rf /mysql/
[root@localhost ~]# find / -name mysql
[root@localhost ~]#
```

删除干净了！接下来就是安装MySQL。

查询是否安装了mariadb

```
1 rpm -qa | grep mariadb
```

卸载mariadb（与MySQL有冲突）

```
1 yum -y remove mariadb*
```

## 2.安装MYSQL

到MySQL官网查找所需的版本，复制下载链接：点击>>[MySQL Community Server \(Archived Versions\)](#)

Product Version:	8.0.27	
Operating System:	Red Hat Enterprise Linux / Oracle Linux	
OS Version:	Red Hat Enterprise Linux 7 / Oracle Linux 7 (x86, 64-bit)	

<b>RPM Bundle</b> (mysql-8.0.27-1.el7.x86_64.rpm-bundle.tar)	Sep 29, 2021	798.6M	<a href="#">Download</a>
MD5: 376f65b6684606a77c46b11eb198a0a5   <a href="#">Signature</a>			
<b>RPM Package, MySQL Server</b> (mysql-community-server-8.0.27-1.el7.x86_64.rpm)	Sep 29, 2021	448.5M	<a href="#">Download</a>
MD5: 457a6c7cd295ae321e5620154202ba56			
<b>RPM Package, Client Utilities</b> (mysql-community-client-8.0.27-1.el7.x86_64.rpm)	Sep 29, 2021	52.6M	<a href="#">Download</a>
MD5: c6c6cdd24a20921e0a4bfcda629089eb			
<b>RPM Package, Client Plugins</b> (mysql-community-client-plugins-8.0.27-1.el7.x86_64.rpm)	Sep 29, 2021	5.7M	<a href="#">Download</a>
MD5: d32adc3e7ec0e543439cf363e3e44314			

## 创建mysql文件夹

```
1 | mkdir mysql
```

## 进入mysql文件夹

```
1 | cd mysql
```

## 提取tar包

```
1 | tar -xvf mysql-8.0.27-1.el7.x86_64.rpm-bundle.tar
```

## 安装common

```
1 | rpm -ivh mysql-community-common-8.0.27-1.el7.x86_64.rpm
```

## 安装libs

```
1 | rpm -ivh mysql-community-libs-8.0.27-1.el7.x86_64.rpm
```

## 安装client

```
1 | rpm -ivh mysql-community-client-8.0.27-1.el7.x86_64.rpm
```

## 安装server

```
1 | rpm -ivh mysql-community-server-8.0.27-1.el7.x86_64.rpm
```

## 查看已经安装的MySQL列表

```
1 | rpm -qa | grep mysql
```

```
[root@localhost mysql]# rpm -qa | grep mysql
mysql-community-libs-8.0.11-1.el7.x86_64
mysql-community-client-8.0.11-1.el7.x86_64
mysql-community-server-8.0.11-1.el7.x86_64
mysql-community-common-8.0.11-1.el7.x86_64
[root@localhost mysql]#
```

## 进行初始化操作

```
1 mysqld --initialize;  
2 chown mysql:mysql /var/lib/mysql -R;  
3 systemctl start mysqld.service;  
4 systemctl enable mysqld
```

## 获取初始化生成的密码

```
1 cat /var/log/mysqld.log | grep password
```

下面的 **=ap5sV6VyOQX** 就是本次初始化生成的密码（划掉那一行是之前安装的，日志没清理 0-0）

以root用户身份登录，使用初始化生成的密码

```
1 mysql -uroot -p
```

更改密码（下面的‘root’是将要设置的密码，换成自己的就行）

```
1 # 修改密码  
2 ALTER USER 'root'@'localhost' IDENTIFIED WITH  
   mysql_native_password BY 'root';  
3 # 刷新权限  
4 FLUSH PRIVILEGES;
```

```
[root@localhost mysql]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.11

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER root@localhost IDENTIFIED BY '123';
Query OK, 0 rows affected (0.04 sec)

mysql>
```

退出MySQL，用新的密码登录，验证是否修改成功

置允许远程登录

```
1 mysql> use mysql
2 mysql> update user set user.Host='%'where user.User='root';
3 mysql> flush privileges;
4 mysql> quit
```

在远程客户端上登录数据库

## 1.7、安装git

yum安装和下载git源码编译安装。但是centos5及以下版本中的yum都没有git，无法使用yum安装，而centos6可以使用yum安装git，但是安装的git是1.7.1版本的，而github需要的git版本最低都不能低于1.7.2。所以如果是centos7及以上版本的，推荐使用yum安装，方便，如果是centos7以下的，请使用git源码编译安装git。

## 17.1 使用yum安装git

```
1 yum -y install git
```

```
正在安装:
git                                x86_64                                1.8.3.1-20.el7                                updates                                4.4 M
为依赖而安装:
perl-Git                           noarch                                1.8.3.1-20.el7                                updates                                55 k

事务概要
-----
安装 1 软件包 (+1 依赖软件包)

总下载量: 4.4 M
安装大小: 22 M
Downloading packages:
(1/2): perl-Git-1.8.3.1-20.el7.noarch.rpm | 55 kB 00:00:00
(2/2): git-1.8.3.1-20.el7.x86_64.rpm      | 4.4 MB 00:00:00
-----
总计                                     5.4 MB/s | 4.4 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
 正在安装   : git-1.8.3.1-20.el7.x86_64                                1/2
 正在安装   : perl-Git-1.8.3.1-20.el7.noarch                          2/2
 验证中    : perl-Git-1.8.3.1-20.el7.noarch                          1/2
 验证中    : git-1.8.3.1-20.el7.x86_64                                2/2

已安装:
git.x86_64 0:1.8.3.1-20.el7

作为依赖被安装:
perl-Git.noarch 0:1.8.3.1-20.el7

完毕！
```

查看是否安装成功

```
1 git --version
```

若出现以上版本号，则代表已经安装了git

## 17.2 使用git源码包安装

### 1、安装依赖的包

```
1 yum -y install curl-devel expat-devel gettext-devel openssl-
  devel zlib-devel gcc perl-ExtUtils-MakeMaker
```

### 2、下载git源码并解压

```
1 $ wget https://github.com/git/git/archive/v2.3.0.zip
2
3 $ unzip v2.3.0.zip
```

### 3、创建一个文件夹，用来安装git

```
1 mkdir /usr/git
```

#### 4、进入git-2.3.0

```
1 cd git-2.3.0
```

#### 5、配置参数，参数中的路径就是3.3步骤创建的那个文件夹的路径

```
1 make prefix=/usr/git/git all
2
3 make prefix=/root/git install
```

#### 6、等待一切安装完成，用 `git --version` 查看版本号，能看到即可

这里可能出现一个问题，如果你之前已经安装过git了，版本比较低，这里展示的可能就是之前的版本，因为系统默认是使用/usr/bin/git下的git，这时候如果想使用你安装的最新版的git，那么操作如下：

6.1) `vim /etc/profile`

6.2)在文件的最后一行加上git的路径

```
1 #git
2 export PATH=/root/git/bin:$PATH
```

6.3) 使文件生效， `source /etc/profile`

6.4)查看版本号， `git --version`，就能看到最新安装的git的版本号

#### 四.如果需要移除git

```
1 yum remove git
```



## 第二章：shell编程

### 2.1 shell简介

#### Shell

Shell 是一个用 C 语言编写的程序，通过 Shell 用户可以访问操作系统内核服务。

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 脚本

Shell 脚本 (shell script) ，是一种为 shell 编写的脚本程序。

业界所说的 shell 通常都是指 shell 脚本，大家要知道，shell 和 shell script 是两个不同的概念。由于习惯的原因，简洁起见，本文出现的 "shell编程" 都是指 shell 脚本编程，不是指开发 shell 自身。

### 2.2 shell 解释器

Shell 编程跟 JavaScript、php 编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。，常见的有： Bourne

Shell (/usr/bin/sh或/bin/sh) Bourne Again Shell (/bin/bash) C

Shell (/usr/bin/csh) K Shell (/usr/bin/ksh) Shell for Root (/sbin/sh) 等

在linux中可以通过cat /etc/shells查看linux系统支持的shell的类型

```
1 [root@localhost bin]# cat /etc/shells
2 /bin/sh
3 /bin/bash
4 /usr/bin/sh
5 /usr/bin/bash
```

本教程关注的是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash 在日常工作中被广泛使用。同时，Bash 也是大多数Linux 系统默认的 Shell。

在一般情况下，人们并不区分 Bourne Shell 和 Bourne Again Shell，所以，像 `#!/bin/sh`，它同样也可以改为 `#!/bin/bash`。

## 2.3 快速入门

### 2.3.1 编写脚本

新建 `/usr/shell/hello.sh` 文件

```
1 #!/bin/bash
2 echo 'hello world'
```

`#!` 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。

`echo` 命令用于向窗口输出文本。

### 2.3.2 执行shell脚本

#### 执行方式1

```
1 [root@centos7-1 shell]# /bin/sh hello.sh
2 hello world
3 [root@centos7-1 shell]# /bin/bash hello.sh
4 hello world
```

问题: `bash` 和 `sh` 是什么关系?

`sh` 是 `bash` 的 快捷方式

```
[root@node04 shells]# ll /bin | grep sh
-rwxr-xr-x. 1 root root 942200 3月 23 2017 bash
-rwxr-xr-x. 1 root root 20760 3月 22 2017 cgsnapshot
lrwxrwxrwx. 1 root root 4 3月 12 16:06 csh -> tcsh
-rwxr-xr-x. 1 root root 109672 10月 17 2012 dash
lrwxrwxrwx. 1 root root 4 3月 12 16:04 sh -> bash
-rwxr-xr-x. 1 root root 388808 3月 22 2017 tcsh 快捷方式
```

## 执行方式2

```
1 [root@centos7-1 shell]# bash hello.sh
2 hello world
3 [root@centos7-1 shell]# sh hello.sh
4 hello world
```

## 执行方式3

```
1 [root@centos7-1 shell]# ./hello.sh
```

## 2.4 变量

在shell脚本中, 定义变量时, 变量名不加美元符号 (\$), 如:

```
username="tom"
```

注意: 变量名和等号之间不能有空格, 这可能和你熟悉的所有编程语言都不一样。同时, 变量名的命名须遵循如下规则:

- 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)。

有效的 Shell 变量名示例如下:

```
1 JACK
2 LD_LIBRARY_PATH
3 _demo
4 var2
```

无效的变量命名：

```
1 ?var=123
2 user*name=runoob
```

## 使用变量

使用一个定义过的变量，只要在变量名前面加美元符号即可，如：

```
1 your_name="tom"
2 echo $your_name
3 echo ${your_name}
```

变量名外面加花括号是可选的，添加花括号是帮助解释器识别变量边界

举例：

```
1 habby="i like ${course}script"
2 habby="i like $coursescript"
```

## 使用语句给变量赋值

举例：

```
1 for file in `ls /etc`
```

或

```
1 for file in $(ls /etc)
```

以上语句是将/etc下目录的文件名循环出来

## 只读变量

使用readonly命令可以将变量定义为只读变量, 只读变量的值不能被改变

```
1 | readonly 变量名
```

举例:

```
1 | username=tom
2 | readonly username
3 | username=jack 报错    /bin/sh: NAME: This variable is read only.
```

## 删除变量

使用unset命令可以删除变量

```
1 | unset variable_name
```

注意:unset命令不能删除只读变量

## 2.5 字符串

字符串是shell编程中最常用最有用的数据类型（除了数字和字符串，也没啥其它类型好用了），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

```
1 skill='java'
2 str='I am goot at $skill'
3 echo $str
```

输出结果为:

```
1 I am goot at $skill
```

单引号字符串的限制:

- 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;
- 单引号字符串中不能出现单独一个的单引号 (对单引号使用转义符后也不行)

## 双引号

```
1 skill='java'
2 str="I am good at $skill"
3 echo $str
```

输出结果为:

```
1 I am good at java
```

双引号的优点:

- 双引号里可以有变量
- 双引号里可以出现转义字符

## 获取字符串长度

```
1 skill='java'
2
3 echo ${skill} # 输出结果: java
4
5 echo ${#skill} # 输出结果: 4
6
```

## 提取子字符串

一下实例从字符串第2个字符开始截取4个字符

```
1 str="i like java"
2
3 echo ${str:2}    #substring(2)
4
5 echo ${str:2:2}  #substring(2,2)
```

## 查找子字符串

查找字符o在那个位置(最先出现的字符)

```
1 str="java is so easy"
2 echo `expr index "$str" o`
```

找的时候是从1开始查找

## 2.6 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

实例

```
1  #-----
2  # 这是一个注释
3  # author: 强哥
4  #-----
5  ##### 用户配置区 开始 #####
6  #
7  #
8  # 这里可以添加脚本描述信息
9  #
10 #
11 ##### 用户配置区 结束 #####
```

如果在开发过程中，遇到大段的代码需要临时注释起来，过一会儿又取消注释，怎么办呢？

每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

### 多行注释

```
1  :<<EOF
2  注释内容...
3  注释内容...
4  注释内容...
5  EOF
```

EOF 也可以使用其他符号:

```
1  :<<!
2  注释内容...
3  注释内容...
4  注释内容...
5  !
```



## 2.7 数组

数组中可以存放多个值。Bash Shell 只支持一维数组（不支持多维数组），初始化时不需要定义数组大小。

与大部分编程语言类似，数组元素的下标由 0 开始。Shell 数组用括号来表示，元素用"空格"符号分割开。

语法格式如下：

### 定义数组方式1

```
1 array_name=(value1 value2 ... valuen)
```

### 定义数组方式2

```
1 array_name[0]=value0
2 array_name[1]=value1
3 array_name[2]=value2
```

### 读取数组格式

```
1 ${array_name[index]}
```

实例：定义数组，并获取数组中内容

```
1 my_array=(A B "C" D)
2
3 echo "第一个元素为：${my_array[0]}"
4 echo "第二个元素为：${my_array[1]}"
5 echo "第三个元素为：${my_array[2]}"
6 echo "第四个元素为：${my_array[3]}"
```

执行脚本：

```
1 $ chmod +x test.sh
2 $ ./test.sh
3 第一个元素为: A
4 第二个元素为: B
5 第三个元素为: C
6 第四个元素为: D
```

## 获取数组中所有元素

使用@ 或 \* 可以获取数组中的所有元素

```
1 my_array[0]=A
2 my_array[1]=B
3 my_array[2]=C
4 my_array[3]=D
5
6 echo "数组的元素为: ${my_array[*]}"
7 echo "数组的元素为: ${my_array[@]}"
```

执行脚本，输出结果如下所示：

```
1 $ chmod +x test.sh
2 $ ./test.sh
3 数组的元素为: A B C D
4 数组的元素为: A B C D
```

## 获取数组的长度

获取数组长度的方法与获取字符串长度的方法相同

```
1 my_array[0]=A
2 my_array[1]=B
3 my_array[2]=C
4 my_array[3]=D
5
6 echo "数组元素个数为: ${#my_array[*]}"
7 echo "数组元素个数为: ${#my_array[@]}"
```

执行脚本，输出结果如下所示：

```
1 $ chmod +x test.sh
2 $ ./test.sh
3 数组元素个数为: 4
4 数组元素个数为: 4
```

## 2.8 参数传递

我们可以在执行 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，1 为执行脚本的第一个参数，2 为执行脚本的第二个参数。

### 实例

以下实例我们向脚本传递三个参数，并分别输出，其中 **\$0** 为执行的文件名（包含文件路径）

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

为脚本设置可执行权限，并执行脚本，输出结果如下所示：

```
1 $ chmod +x test.sh
2 $ ./test.sh 1 2 3
3 Shell 传递参数实例!
4 执行的文件名: ./test.sh
5 第一个参数为: 1
6 第二个参数为: 2
7 第三个参数为: 3
```

另外，还有几个特殊字符用来处理参数：

参数处理	说明
<code>\$#</code>	传递到脚本的参数个数
<code>*</code>   以一个单字符串显示所有向脚本传递的参数。如 <code>// *"</code> 用"括起来的情况、以 <code>"12 ... \$n"</code> 的形式输出所有参数。	
<code>\$\$</code>	脚本运行的当前进程ID号
<code>\$!</code>	后台运行的最后一个进程的ID号
<code>@</code>   与 <code>*</code> 相同，但是使用时加引号，并在引号中返回每个参数。如 <code>"@//</code> 用 <code>//</code> 括起来的情况、以 <code>//1" "2// ...//n"</code> 的形式输出所有参数。	
<code>\$-</code>	显示Shell使用的当前选项，与 <code>set</code> 命令功能相同。
<code>\$?</code>	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。

## 实例

```
1 echo "Shell 传递参数实例! ";
2 echo "第一个参数为: $1";
3
4 echo "参数个数为: $#";
5 echo "传递的参数作为一个字符串显示: $*";
```

执行脚本，输出结果如下所示：

```
1 $ chmod +x test.sh
2 $ ./test.sh 1 2 3
3 Shell 传递参数实例!
4 第一个参数为: 1
5 参数个数为: 3
6 传递的参数作为一个字符串显示: 1 2 3
```

## 2.9 运算符

Shell 和其他编程语言一样，支持多种运算符，今天我们主要学习两类运算符：

- 算数运算符
- 关系运算符

原生bash不支持简单的数学运算，但是可以通过其他命令来实现,比如：

`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。

例如，两个数相加(注意使用的是反引号 ``` 而不是单引号 `'`)：

举例：两个数相加

```
1 val=`expr 1 + 2`
2 echo $val
```

注意：

- 1 1) 表达式和运算符之间要有空格, 例如 `2+2` 是不对的, 必须写成 `2 + 2`, 这与我们熟悉的大多数编程语言不一样。
- 2 2) 完整的表达式要被 ``` 包含, 注意这个字符不是常用的单引号, 在 `Esc` 键下边。

### 2.9.1 算数运算符

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

注意: 条件表达式要放在方括号之间, 并且要有空格, 例如: `[ $a==$b ]` 是错误的,

必须写成 `[ $a == $b ]`。

## 案例

```
1  #!/bin/bash
2  a=4
3  b=20
4
5  #加法运算
6  echo `expr $a + $b`
7
8  #减法运算
9  echo `expr $a - $b`
10
11 #乘法运算，注意*号前面需要反斜杠
12 echo `expr $a \* $b`
13
14 #除法运算
15 echo ``$a / $b`
16 此外，还可以通过 (( ))、$(())、$[ ] 进行算术运算。
17 ((a++))
18 echo "a = $a"
19 c=$((a + b))
20 d=$((a + b))
21 echo "c = $c"
22 echo "d = $d"
```

### 2.9.2 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。  
下表列出了常用的关系运算符，假定变量 a 为 10，变量 b 为 20：

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

## 举例

```
1 a=10
2 b=20
3 if [ $a -eq $b ]
4 then
5     echo "$a -eq $b : a 等于 b"
6 else
7     echo "$a -eq $b: a 不等于 b"
```

## 2.10 echo命令

Shell 的 echo 是用于字符串的输出。命令格式：

```
1 echo string
```



您可以使用echo实现更复杂的输出格式控制。

### 1.显示普通字符串:

```
1 echo "It is a test"
```

这里的双引号完全可以省略，以下命令与上面实例效果一致：

```
1 echo It is a test
```

### 2.显示转义字符

```
1 echo "\"It is a test\""
```

结果将是：

```
1 "It is a test"
```

同样，双引号也可以省略

### 3.显示变量

read 命令从标准输入中读取一行,并把输入行的每个字段的值指定给 shell 变量

```
1 #!/bin/sh
2 read name
3 echo "$name It is a test"
```

以上代码保存为 test.sh，name 接收标准输入的变量，结果将是：

```
1 [root@www ~]# sh test.sh
2 OK                      #标准输入
3 OK It is a test         #输出
```

### 4.显示换行

```
1 echo -e "OK! \n" # -e 开启转义
2 echo "It is a test"
```

输出结果：

```
1 OK!  
2  
3 It is a test
```

## 5.显示不换行

```
1 #!/bin/sh  
2 echo -e "OK! \c" # -e 开启转义 \c 不换行  
3 echo "It is a test"
```

输出结果：

```
1 OK! It is a test
```

## 6.显示结果定向至文件

```
1 echo "It is a test" > myfile
```

## 7.原样输出字符串，不进行转义或取变量(用单引号)

```
1 echo '$name\"'
```

输出结果：

```
1 $name\"'
```

## 2.11 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。

### 1.数值测试

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

## 实例

```
1 num1=100
2 num2=100
3 if test ${num1} -eq ${num2}
4 then
5     echo '两个数相等!'
6 else
7     echo '两个数不相等!'
8 fi
```

输出结果：

```
1 两个数相等!
```

## 2.12 流程控制

### 2.12.1 if语句

- if语句格式

```
1  if condition
2  then
3      command1
4      command2
5      ...
6      commandN
7  fi
```

- if.else语句

```
1  if condition
2  then
3      command1
4      command2
5      ...
6      commandN
7  else
8      command
9  fi
```

- if..elif..else

```
1  if condition1
2  then
3      command1
4  elif condition2
5  then
6      command2
7  else
8      commandN
9  fi
```

## 案例

```
1  [root@hadoop01 export]# cat if_test.sh
```

```
2  #!/bin/bash
3  a=20
4  b=10
5  # 需求1: 判断 a 是否 100
6  if [ $a > 100 ]; then
7  echo "$a 大于 100"
8  fi
9
10 # 需求2: 判断 a 是否等于 b
11 if [ $a -eq $b ]; then
12 echo "$a 等于 $b"
13 else
14 echo "$a 不等于 $b"
15 fi
16
17 # 需求3: 判断 a 与 b 比较
18 if [ $a -lt $b ]; then
19 echo "$a 小于 $b"
20 elif [ $a -eq $b ]; then
21 echo "$a 等于 $b"
22 else
23 echo "$a 大于 $b"
24 fi
25
26 # 需求4: 判断 (a + 10) 和 (b * b) 比较大小
27 if test $[ a + 10 ] -gt $[ b * b ]; then
28 echo "(a+10) 大于 (b * b)"
29 else
30 echo "(a+10) 小于或等于 (b*b)"
31 fi
```

## 2.12.2 for 循环

### 格式

```
1 for variable in (list); do
2     command
3     command
4     ...
5 done
```

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。

例如，顺序输出当前列表中的数字：

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

顺序输出字符串中的字符：

```
1 #!/bin/bash
2
3 for str in This is a string
4 do
5     echo $str
6 done
```

### 练习

```
1 # 需求1: 遍历 1~5
2 # 需求2: 遍历 1~100
3 # 需求3: 遍历 1~100之间的奇数
4 # 需求4: 遍历 根目录 下的内容
```

```
1 [root@hadoop01 export]# cat for_test.sh
2 #!/bin/bash
3
4 # 需求1: 遍历 1~5
5 for i in 1 2 3 4 5; do
6 echo $i;
7 done
8
9 # 需求2: 遍历 1~100
10 for i in {1..100}; do
11 echo $i
12 done
13
14 # 需求3: 遍历 1~100之间的奇数
15 for i in {1..100..2}; do
16 echo $i
17 done
18
19 # 需求4: 遍历 根目录 下的内容
20 for f in `ls /`; do
21 echo $f
22 done
```

### 2.12.3 while 语句

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。其格式为

```
1 while condition; do
2     command
3 done
```

需求: 计算 1~100 的和

```
1 #!/bin/bash
2
3 sum=0
4 i=1
5 while [ $i -le 100 ]; do
6     sum=$(( sum + i ))
7     i=$(( i + 1 ))
8 done
9 echo $sum
```

运行脚本，输出：

```
1 5050
```

## 2.12.4 case

**case ... esac** 为多选择语句，与其他语言中的 **switch ... case** 语句类似，是一种多分支选择结构，每个 **case** 分支用右圆括号开始，用两个分号 **;;** 表示 **break**，即执行结束，跳出整个 **case ... esac** 语句，**esac**（就是 **case** 反过来）作为结束标记。

```
1 case 值 in
2     模式1)
3         command1
4         command2
5         ...
6         commandN
7     ;;
8 模式2)
```



```

9      command1
10     command2
11     ...
12     commandN
13     ;;
14 esac

```

case 工作方式如上所示，取值后面必须为单词 **in**，每一模式必须以右括号结束。取值可以为变量或常数，匹配发现取值符合某一模式后，其间所有命令开始执行直至 **;;**。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用 **\***号 捕获该值，再执行后面的命令。

下面的脚本提示输入1到4，与每一种模式进行匹配：

```

1  echo '输入 1 到 4 之间的数字:'
2  read aNum
3  case $aNum in
4      1) echo '你选择了 1'
5          ;;
6      2) echo '你选择了 2'
7          ;;
8      3) echo '你选择了 3'
9          ;;
10     4) echo '你选择了 4'
11         ;;
12     *) echo '你没有输入 1 到 4 之间的数字'
13         ;;
14 esac

```

输入不同的内容，会有不同的结果，例如：

```

1  输入 1 到 4 之间的数字:
2  你输入的数字为:
3  3
4  你选择了 3

```

## 2.13 跳出循环

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

### 2.13.1 break命令

break命令允许跳出所有循环（终止执行后面的所有循环）。

需求: 执行死循环 每隔1秒打印当前时间, 执行10次停止

```
1  #!/bin/bash
2  # 需求：执行死循环 每隔1秒打印当前时间，执行10次停止
3  i=0;
4  while true; do
5      sleep 1
6      echo $i `date +%Y-%m-%d %H:%M:%S`
7      i=$(( i + 1 ))
8      if [ $i -eq 10 ]; then
9          break
10     fi
11 done
```

### 2.13.2 continue

continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

需求: 打印 1~30, 注意 跳过3的倍数

```
1 #!/bin/bash
2 # 需求：打印 1~30，注意 跳过3的倍数
3 for i in {1..30}; do
4     if test $[ i % 3 ] -eq 0; then
5         continue
6     fi
7     echo $i
8 done
```

## 2.14 函数使用

### 2.14.1 函数的快速入门

#### 格式

```
1 [ function ] funname()
2 {
3     action;
4     [return int;]
5 }
```

- 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)

#### 快速入门

```
1 #!/bin/bash
2 demoFun(){
3     echo "这是我的第一个 shell 函数!"
4 }
5 echo "-----函数开始执行-----"
6 demoFun
7 echo "-----函数执行完毕-----"
```

```

1 funWithReturn(){
2     echo "这个函数会对输入的两个数字进行相加运算..."
3     echo "输入第一个数字: "
4     read aNum
5     echo "输入第二个数字: "
6     read anotherNum
7     echo "两个数字分别为 $aNum 和 $anotherNum !"
8     return $(( $aNum+$anotherNum ))
9 }
10 funWithReturn
11 echo "输入的两个数字之和为 $? !"

```

## 2.14.2 传递参数给函数

在Shell中，调用函数时可以向其传递参数。在函数体内部，通过 *n* 的形式来获取参数的值，例如，1表示第一个参数，\$2表示第二个参数...

带参数的函数示例：

```

1 #!/bin/bash
2 funWithParam(){
3     echo "第一个参数为 $1 !"
4     echo "第二个参数为 $2 !"
5     echo "第十个参数为 $10 !"
6     echo "第十个参数为 ${10} !"
7     echo "第十一个参数为 ${11} !"
8     echo "参数总数有 $# 个!"
9     echo "作为一个字符串输出所有参数 $* !"
10 }
11 funWithParam 1 2 3 4 5 6 7 8 9 34 73

```

输出结果：

```
1 第一个参数为 1 !
2 第二个参数为 2 !
3 第十个参数为 10 !
4 第十个参数为 34 !
5 第十一个参数为 73 !
6 参数总数有 11 个!
7 作为一个字符串输出所有参数 1 2 3 4 5 6 7 8 9 34 73 !
```

## 2.15 shell编程案例

### 2.15.1 统计目录文件

编写脚本/root/bin/sumfile.sh，统计/etc，/var，/usr目录中共有多少个一级子目录和文件

```
1 #!/bin/bash
2 ##写一个脚本/root/bin/sumfile.sh,统计/etc, /var, /usr目录中共有多少
   个一级子目录和文件
3 num1=$(ls -l /etc | wc -l)
4 num2=$(ls -l /var | wc -l)
5 num3=$(ls -l /usr | wc -l)
6 let Num=num1+num2+num3
7 echo $Num
```

### 2.15.2 猜数字小游戏

脚本生成一个 100 以内的随机数字，提示用户猜数字，根据用户的输入，提示用户是否猜对，

猜大或猜小继续循环猜，猜对跳出循环。

RANDOM 为系统自动的系统变量，值为 0-32767 的随机数

```
1 #!/bin/bash
```

```

2  #猜数字小游戏
3  num=$((RANDOM%100+1))
4
5  while :
6  do
7      read -p "计算机生成一个1-100的随机数" cat
8      if [ $cat -eq $num ];then
9          echo "猜对了"
10         break
11     elif [ $cat -le $num ];then
12         echo "小了"
13     else
14         echo "大了"
15     fi
16 done
17

```

### 2.15.3 使用 case、while 和 read 进行查看系统信息

编写一个交互脚本，执行脚本时候键盘输入对应的参数，脚本反馈对应信息，当输入

为“exit”时候脚本退出。脚本采用 read 捕捉键盘输入信息，然后使用 case 进行条件选择并执行命令，最

后使用 while 对 case 语句进行循环。

```

1  #!/bin/bash
2  #EOF (End Of File)表示文本结束符。注意：EOF必须顶行写，前面不能用制表符或者空格。
3  #cat << EOF的意思是以EOF输入字符为标准输入结束，就是当你输入cat << EOF的时候，你可以随意输入字符，但是当输入EOF#的时候就结束了
4  cat << EOF
5  D|d 显示硬盘信息
6  M|m 显示内存信息
7  S|s 显示交换分区信息

```

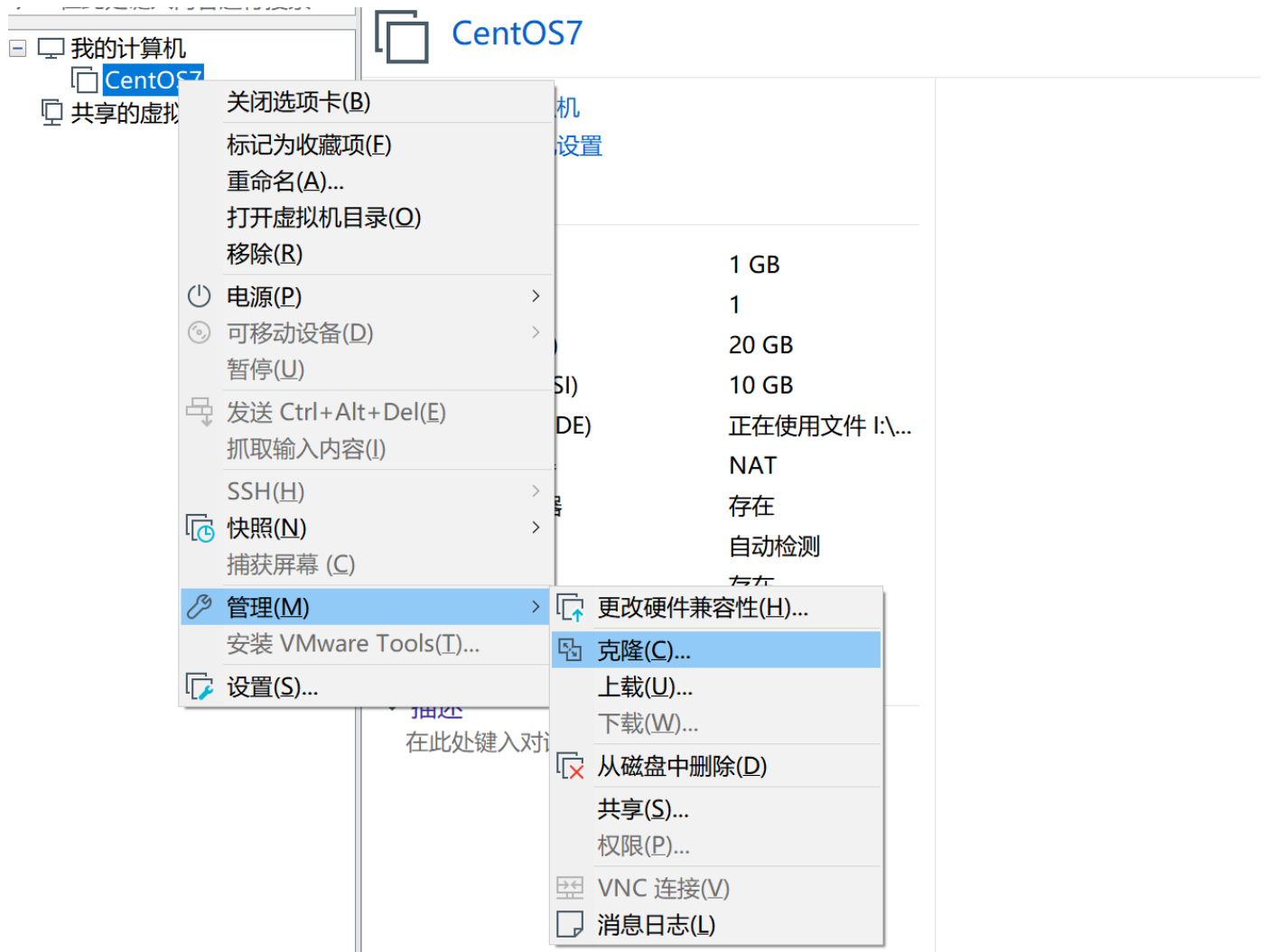
```
8 EOF
9
10 read -p "请输入以上对应参数:" CANSHU
11 while [ $CANSHU != "exit" ]; do
12 case $CANSHU in
13 d|D)
14     df -h
15     ;;
16 m|M)
17     free -m | grep Mem
18     ;;
19 s|S)
20     free -m | grep Swap
21     ;;
22 *)
23     echo "Ukown"
24     ;;
25 esac
26 read -p "请再次输入以上对应参数:" CANSHU
27 done
```

## 第三章：集群搭建

### 3.1、克隆节点

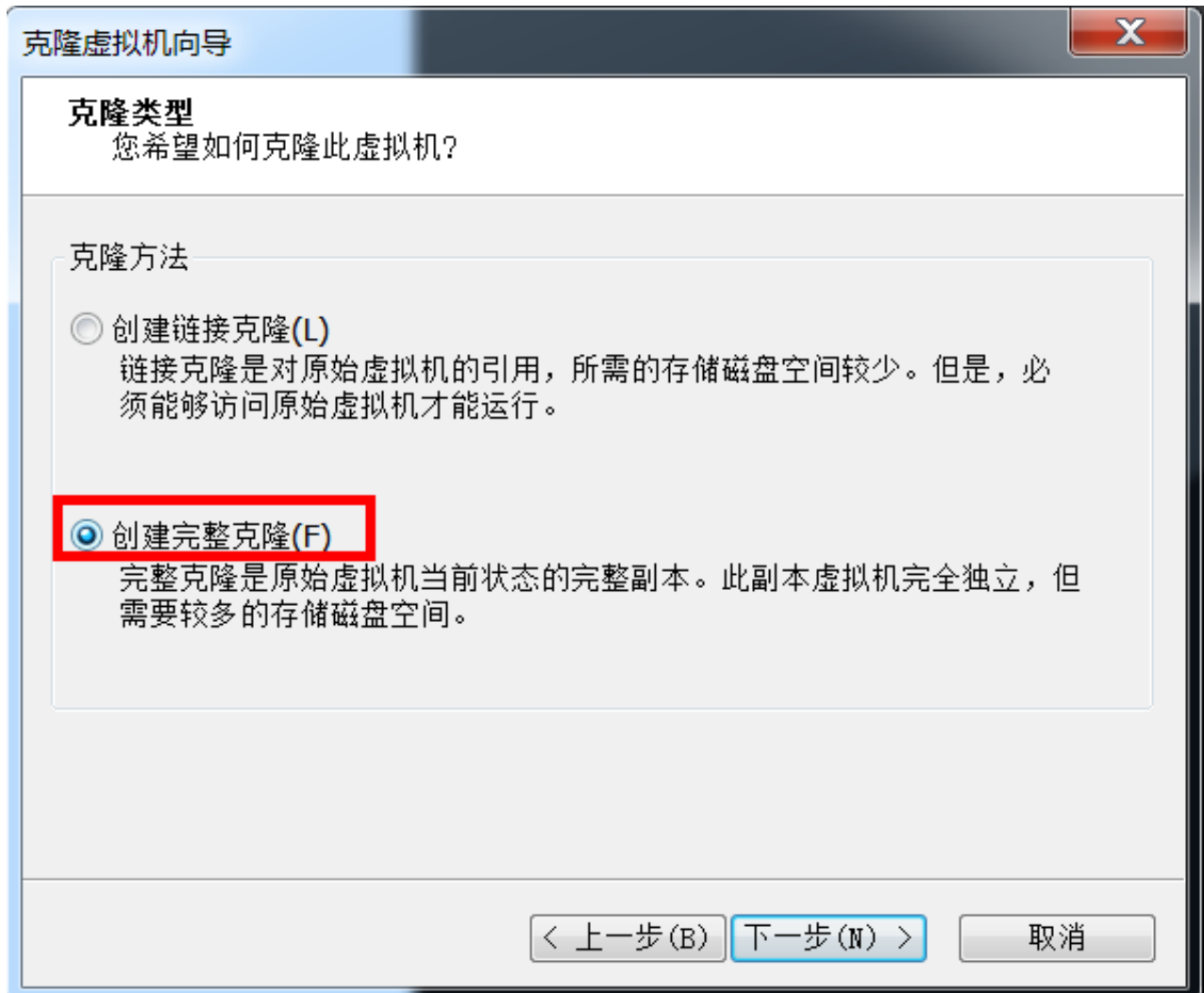
关闭当前centos7

右键点击克隆









克隆虚拟机向导

X

新虚拟机名称

您希望该虚拟机使用什么名称？

虚拟机名称(V)

CentOS 7-2

位置(L)

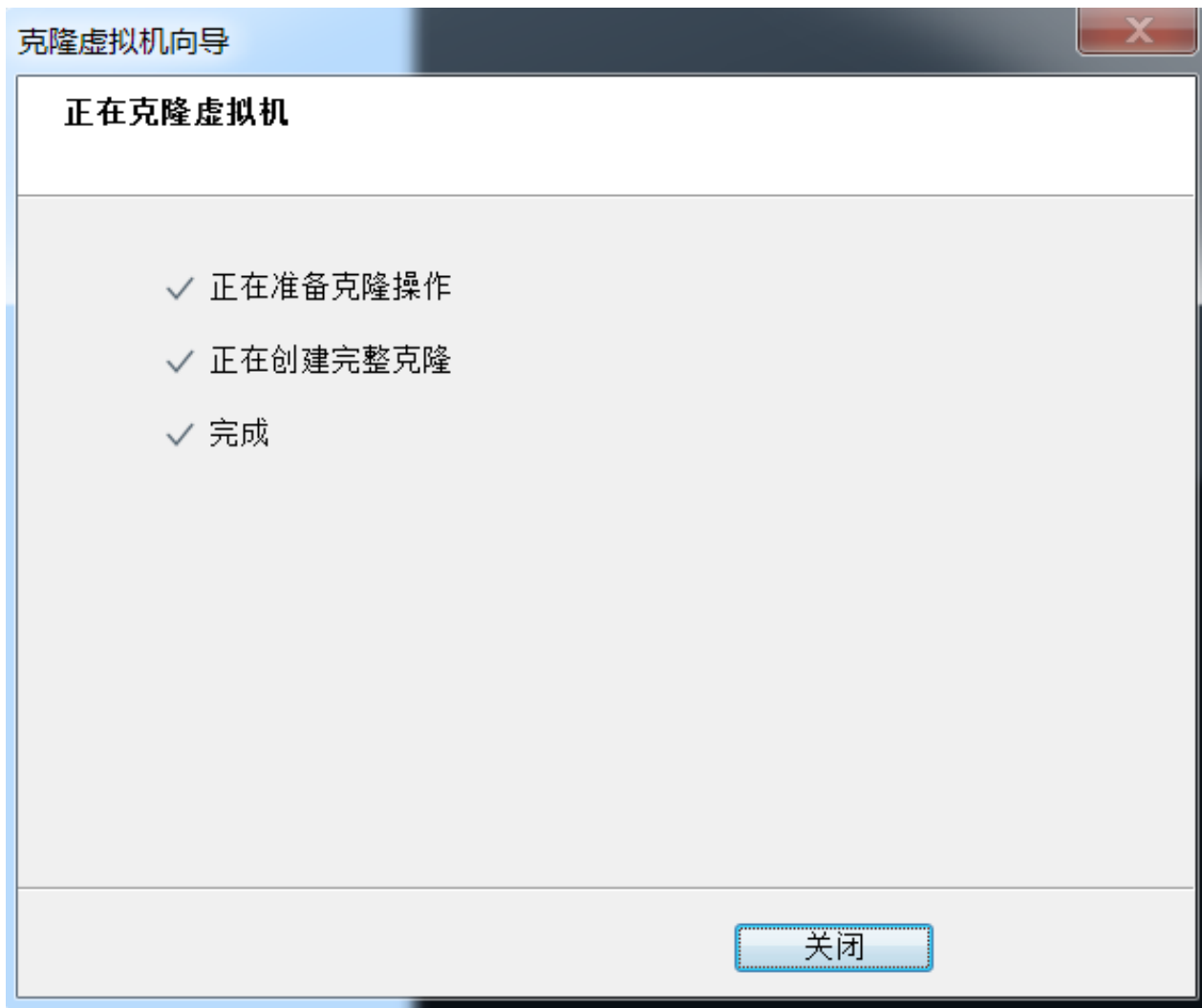
E:\developer\centos7-2

浏览(R)...

< 上一步(B)

完成

取消



### 3.2、配置克隆后机器

- 网络配置文件ifcfg-ens33的ip地址

```
1 vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=a9bb20aa-c5d5-4a88-a3d2-b3f4434b8e62
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.80.102
NETMASK=255.255.255.0
GATEWAY=192.168.80.2
DNS1=8.8.8.8
~
~
```

## 重启网卡服务器

```
1 | systemctl restart network
```

## 检查ip地址是否变更

```
1 | ip addr
```

```
"/etc/sysconfig/network-scripts/ifcfg-ens33" 19L, 360C written
[root@localhost ~]# systemctl network restart
Unknown operation 'network'.
[root@localhost ~]# systemctl restart network
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:74:86:e0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.80.102/24 brd 192.168.80.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::8def:aa11:5c16:3f1c/64 scope link
        valid_lft forever preferred_lft forever
    inet6 fe80::7d54:9b08:69fc:d04d/64 scope link tentative dadfailed
        valid_lft forever preferred_lft forever
[root@localhost ~]# _
```

```
[root@localhost ~]# ping www.baidu.com
PING www.wshifen.com (103.235.46.39) 56(84) bytes of data:
64 bytes from 103.235.46.39 (103.235.46.39): icmp_seq=1 ttl=128 time=393 ms
64 bytes from 103.235.46.39 (103.235.46.39): icmp_seq=2 ttl=128 time=365 ms
64 bytes from 103.235.46.39 (103.235.46.39): icmp_seq=3 ttl=128 time=438 ms
64 bytes from 103.235.46.39 (103.235.46.39): icmp_seq=4 ttl=128 time=418 ms
64 bytes from 103.235.46.39 (103.235.46.39): icmp_seq=5 ttl=128 time=384 ms
^C
--- www.wshifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 365.827/400.089/438.186/25.632 ms
[root@localhost ~]# _
```

- hostname主机名，不要跟源克隆机一致

查看主机名命令

```
1 hostnamectl
```

更改主机名

```
1 hostnamectl set-hostname xxx
```

重启机器 `reboot`

## 3.2、关闭selinux

安全增强型 Linux (Security-Enhanced Linux) 简称 SELinux，它是一个 Linux 内核模块，也是 Linux 的一个安全子系统。

linux操作系统中默认开启 SELinux也处于启动状态，一般状态为enforcing。致使很多服务端口默认是关闭的。所以好多服务初学者明明配置文件正确，等验证时有时连ping也ping不通或者导致SSH连接异常。

如何关闭

```
1 vi /etc/selinux/config
```

将文件中的SELINUX="" 为 disabled，然后重启。

### 3.3、ssh 安全外壳协议

SSH 叫安全外壳协议（Secure Shell），是一种加密的网络传输协议，可在不安全的网络中网络服务提供安全的传输环境。它通过在网络中创建安全隧道来实现 SSH 客户端和服务端之间的连接。

#### 基本的用法

SSH 主要用于远程登录。假定你要以用户名 `user`，登录远程主机 `host`，只要一条简单命令就可以了。

```
1 $ ssh user@host
```

如果本地用户名与远程用户名一致，登录时可以省略用户名。

```
1 $ ssh host
```

SSH 的默认端口是22，也就是说，你的登录请求会送进远程主机的22端口。使用 `p` 参数，可以修改这个端口。

```
1 $ ssh -p 2222 user@host
```

上面这条命令表示，`ssh` 直接连接远程主机的 `2222` 端口

#### ssh工作原理

SSH的安全性比较好，其对数据进行加密的方式主要有两种：对称加密（密钥加密）和非对称加密（公钥加密）。

#### 对称加密

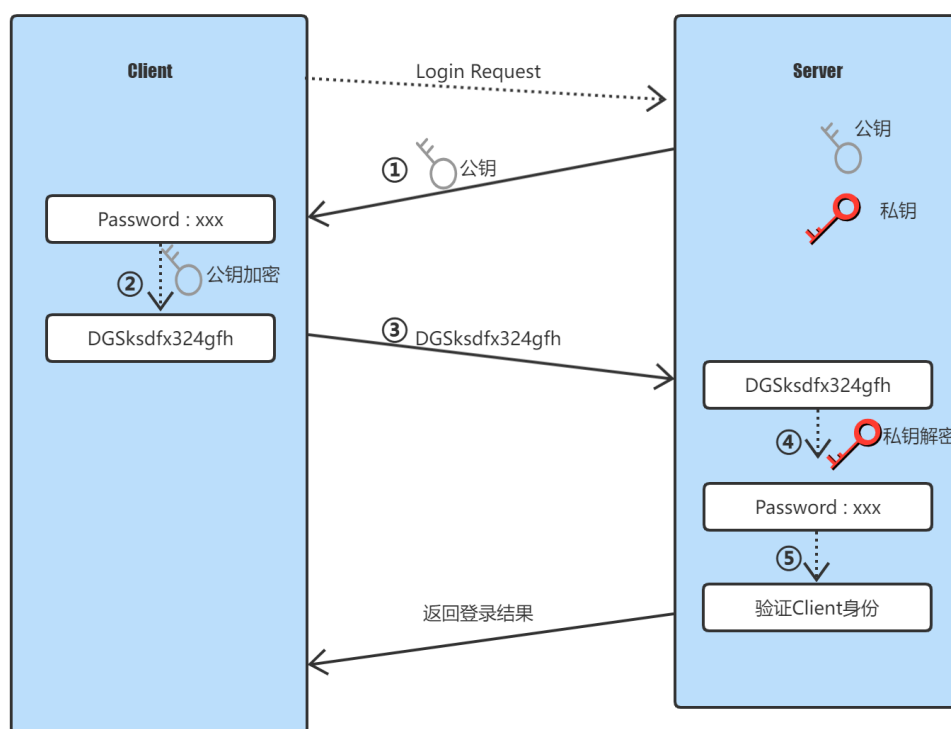
加密解密使用的是同一套密钥。Client端把密钥加密后发送给Server端，Server用同一套密钥解密。对称加密的加密强度比较高，很难破解,但是，Client数量庞大，很难保证密钥不泄漏。如果有一个Client端的密钥泄漏，那么整个系统的安全性就存在严重的漏洞。



## 非对称加密

由于对称加密的这个弊端，产生了非对称加密，非对称加密中有两个密钥：公钥和私钥。公钥由私钥产生，但却无法推算出私钥；公钥加密后的密文，只能通过对应的私钥来解密。

非对称加密的登录流程：



初始状态：client 终端要登录 Server 服务器，发起连接请求 `ssh work@server.com`

1. 服务端运行有 `ssh` 服务，并持续监听 22 号端口，因此可以生成一对公钥和私钥；此时将公钥返回给客户端
2. 客户端使用公钥，对登录密码进行加密，（如服务器 `work` 用户密码为 `xxx`），



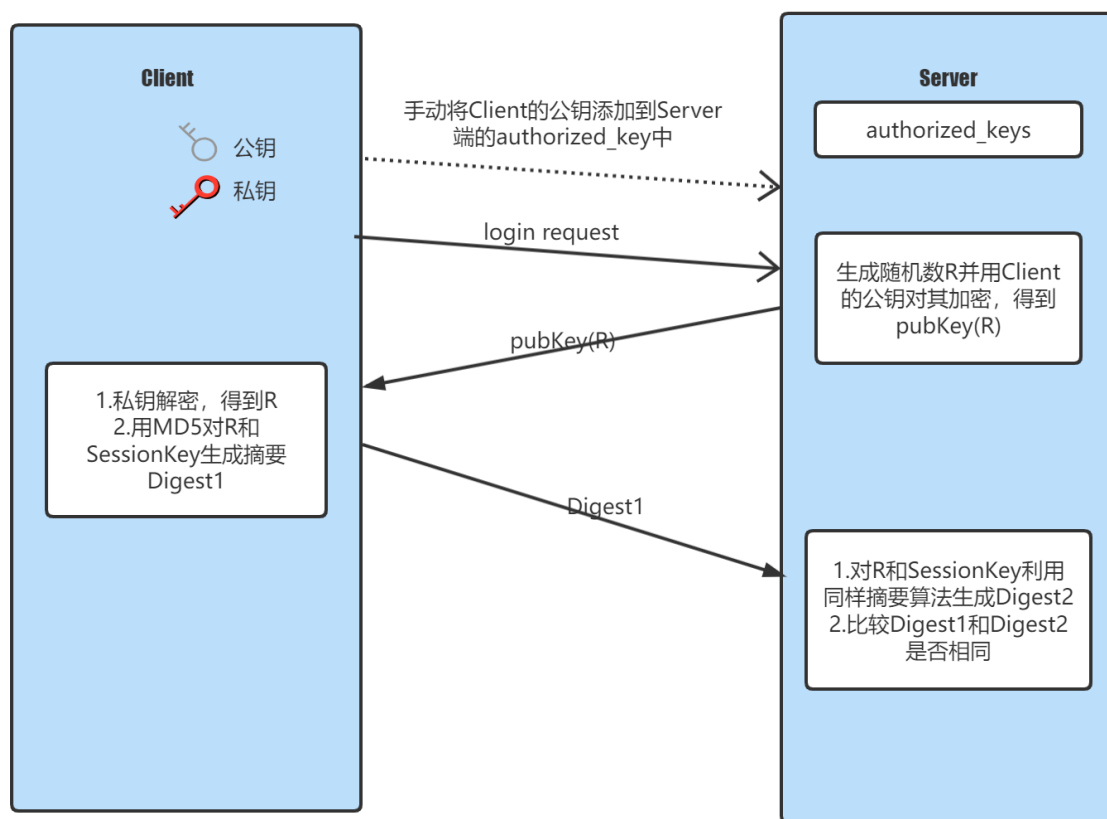
## 生成公钥加密字符串

3. 客户端将公钥加密字符串发送给服务端
4. 服务端使用私钥，解密公钥加密字符串，得到原始密码
5. 校验密码是否合法（此为本机 work 密码）
6. 返回登录结果给客户端：成功登录或密码错误

在非对称加密中，由于只有公钥会被传输，而私钥是服务端本地保存，因此即便公钥被监听，也无法拿到原始密码，从而登录服务器。

## 3.4、ssh 免密登录

我们已经掌握如何使用 `ssh` 登录远程服务器了，但是每次登录都要输入密码，比较麻烦。`ssh` 提供一种免密登录的方式：公钥登录。



1. 在客户端使用 `ssh-keygen` 生成一对密钥：公钥+私钥
2. 将客户端公钥追加到服务端的 `authorized_key` 文件中，完成公钥认证操作
3. 认证完成后，客户端向服务端发起登录请求，并传递公钥到服务端
4. 服务端检索 `authorized_key` 文件，确认该公钥是否存在
5. 如果存在该公钥，则生成随机数 `R`，并用公钥来进行加密，生成公钥加密字符

串 `pubKey(R)`

6. 将公钥加密字符串传递给客户端
7. 客户端使用私钥解密公钥加密字符串，得到 `R`
8. 服务端和客户端通信时会产生一个会话 `ID(sessionKey)`，用 `MD5` 对 `R` 和 `SessionKey` 进行加密，生成摘要（即 `MD5` 加密字符串）
9. 客户端将生成的 `MD5` 加密字符串传给服务端
10. 服务端同样生成 `MD5(R, SessionKey)` 加密字符串
11. 如果客户端传来的加密字符串等于服务端自身生成的加密字符串，则认证成功
12. 此时不用输入密码，即完成建连，可以开始远程执行 `shell` 命令了

## 实现免密登录

### 1.配置hostname与IP映射

分别的在centos7-1 centos7-2 centos7-3中配置

```
1 vi /etc/hosts
```

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.80.100 centos7-1
192.168.80.101 centos7-2
192.168.80.102 centos7-3
```

### 2.免密登录步骤

第一步：在三台机器器执行行以下命令，生成公钥与私钥

```
1 ssh-keygen -t rsa #在centos7-1和centos7-2和centos7-3上面都要执行，产生公钥和私钥
```

第二步：将centos7-2和centos7-3的公钥拷贝到centos7-1

```
1 ssh-copy-id centos7-1 #在centos7-1 ,centos7-2和centos7-3上执行
```

第三步：再将centos7-1的公钥分发给centos7-2和centos7-3

```
1 scp authorized_keys centos7-2:$PWD
2 scp authorized_keys centos7-3:$PWD
```

第四步：使用ssh完成免密登录即可

```
1 ssh centos7-2    #其他节点登录centos7-2
```

## 第四章：定时任务与时钟同步

### 4.1 crontab 任务调度

#### 4.1.1 概念及基本语法

crontab 进行 定时任务的设置

基本语法

crontab [选项]

选项	英文	含义
-e	edit	编辑crontab定时任务
-l	displayed	显示crontab任务
-r	remove	删除当前用户所有的crontab任务

#### 4.1.2 参数细节说明

5个占位符的说明

```
[root@localhost log]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name  command to be executed

[root@localhost log]#
```

占位符	含义	范围
第一个 *	一小时当中的第几分钟	0~59
第二个 *	一天当中的第几小时	0~23
第三个 *	一个月当中的第几天	1~31
第四个 *	一年当中的第几月	1~12
第五个 *	一周当中的星期几	0~7(0和7都代表星期日)

举例：

命令	含义
* * * * * command	实例1： 每1分钟执行一次command 3,15
3,15 * * * * command	实例2： 每小时的第3和第15分钟执行
3,15 8-11 * * * command	实例3： 在上午8点到11点的第3和第15分钟执行
3,15 8-11 */2 * * command	实例4： 每隔两天的上午8点到11点的第3和第15分钟执行
3,15 8-11 * * 1 command	实例5： 每个星期一的上午8点到11点的第3和第15分钟执行
30 21 * * * /etc/init.d/smb restart	实例6： 每晚的21:30重启smb
45 4 1,10,22 * * /etc/init.d/smb restart	每月1、10、22日的4:45重启smb
10 1 * * 6,0 /etc/init.d/smb restart	实例8： 每周六、周日的1:10重启smb
0,30 18-23 * * * /etc/init.d/smb restart	每天18:00至23:00之间每隔30分钟重启smb

### 4.1.3 任务调度案例

案例: 每隔1分钟将时间打印到 /usr/test/mydate1.txt 文件中

第一步: `date >> /usr/test/mydate1.txt` 测试命令

第二步: 通过 `crontab -e` 进入 定时任务

第三步: 编辑定时任务命令

```
1 */1 * * * * date >> /usr/test/mydate1.txt
```

## 第四步: 检测是否成功

```
1 | cd /usr/test/ && tail -f mydate1.txt
```

### 案例2: 脚本方式定时任务打印时间

## 第一步: 书写测试命令

```
1 | echo `date +%Y-%m-%d %H:%M:%S` >> mydate2.txt
```

## 第二步: 将命令放到脚本中 vim /export/task.sh

```
1 | echo `date +%Y-%m-%d %H:%M:%S` >> /usr/test/mydate2.txt
```

## 第三步: 增加可执行权限

```
1 | chmod +x /usr/test/task.sh
```

## 第四步: 执行脚本

```
1 | /usr/test/task.sh
```

## 第五步: 进入 定时任务 crontab -e

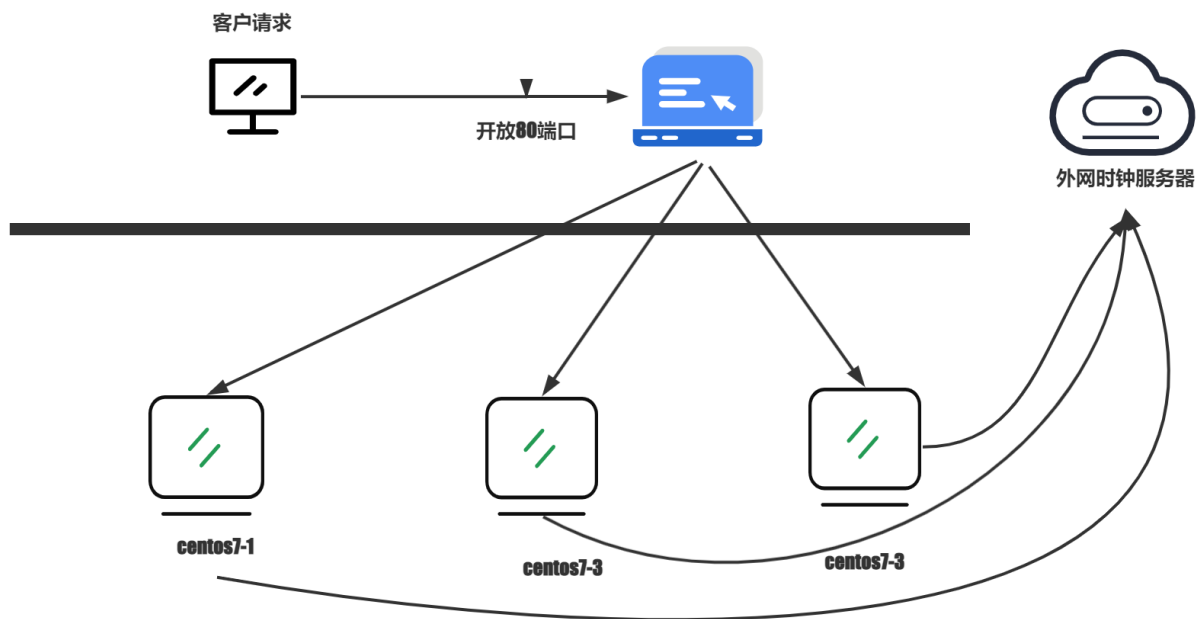
```
1 | */1 * * * * date >> /usr/test/mydate1.txt  
2 | */1 * * * * /usr/test/task.sh
```

## 第六步: 测试 观察结果

```
1 | tail -f /usr/test/mydate2.txt
```

## 4.2 时钟服务器

通过命名和时钟服务器同步时间:



### 时钟服务器

NTP是网络时间协议（Network Time Protocol）的简称，就是通过网络协议使计算机之间的时间同步化。

#### 1.安装ntp服务

```
1 yum -y install ntp ntpdate
```

#### 2.查看ntp服务是否启动

```
1 systemctl is-enabled ntpdate
```

```
[root@centos7-1 test]# systemctl is-enabled ntpdate
disabled
```

### 3. 设置为开机启动

```
1 | systemctl enable ntpdate
```

```
[root@centos7-1 test]# systemctl is-enabled ntpdate
enabled
```

### 4. 根据ntpdate设置时钟同步服务器

网络计时协议 (NTP)

```
1 | ntpdate us.pool.ntp.org
```

阿里云时钟同步服务器

```
1 | ntpdate ntp4.aliyun.com
```

三台机器定时任务：直接与阿里云服务器进行时钟同步

```
1 | crontab -e
2 | */1 * * * * /usr/sbin/ntpdate ntp4.aliyun.com
```

千锋教育Java教研院 关注公众号【Java架构栈】 下载所有课程代码课件及工具 让技术回归本该有的纯净!