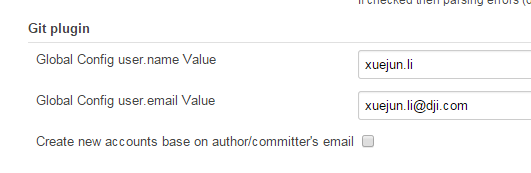
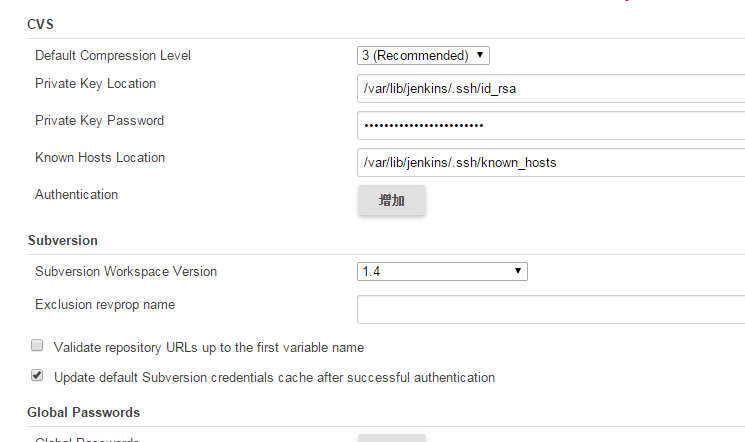
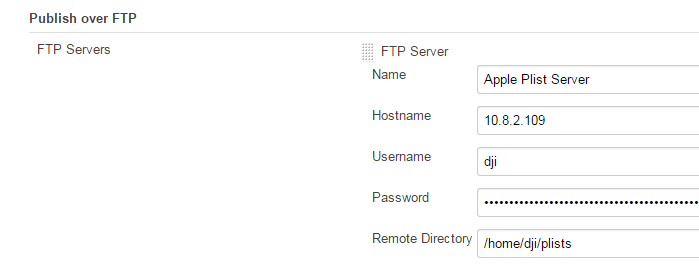
# 脚本

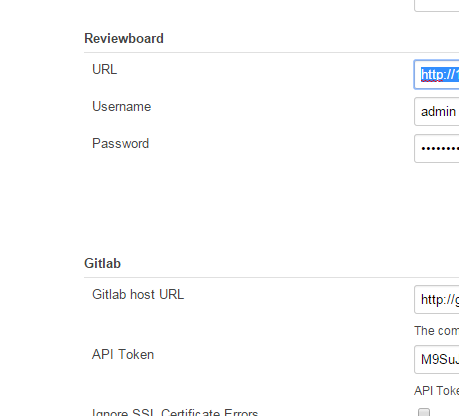
## Apk解压脚本

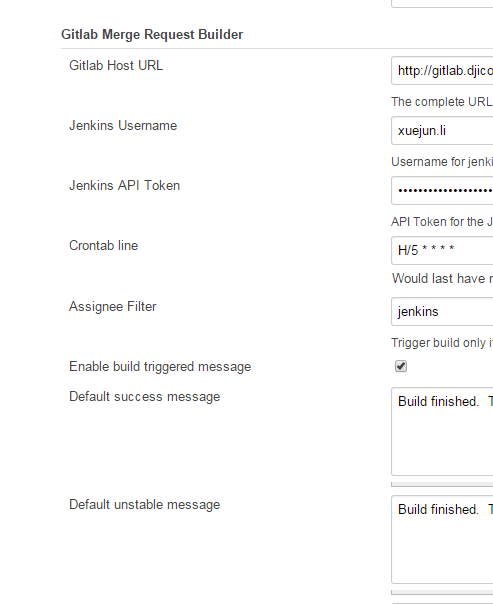
<http://www.runoob.com/linux/linux-file-attr-permission.html>











# 脚本原理

# android的ABI/CPU

## Android中的ABI

* ABI是Application Binary Interface的缩写。
* ABI常表示两个程序模块之间的接口，且其中一个模块常为机器码级别的library或操作系统。
* ABI定义了函数库的调用、应用的二进制文件（尤其是.so）如何运行在相应的系统平台上等细节。
* Android目前支持以下七种ABI：**armeabi、armeabi-v7a、arm64-v8a、x86、x86\_64、mips、mips64**。

## Android中的CPU

* 不同的Android设备使用支持不同指令集的CPU。
* Android目前有以下七种cpu架构：**ARMv5、ARMv7、ARMv8、x86、x86\_64、MIPS和MIPS64**。

## Android中的ABI与CPU

* 每种CPU架构都有其自己支持的ABIs。可通过Build.SUPPORTED\_ABIS得到根据偏好排序的设备支持的ABI列表。

| **CPU（纵向）/ABI（横向）** | **armeabi** | **armeabi-v7a** | **arm64-v8a** | **x86** | **x86\_64** | **mips** | **mips64** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ARMv5 | 支持(1) |  |  |  |  |  |  |
| ARMv7 | 支持(2) | 支持(1) |  |  |  |  |  |
| ARMv8 | 支持(3) | 支持(2) | 支持(1) |  |  |  |  |
| x86 | 支持(3) | 支持(2) |  | 支持(1) |  |  |  |
| x86\_64 | 支持(4) | 支持(3) |  | 支持(2) | 支持(1) |  |  |
| MIPS |  |  |  |  |  | 支持(1) |  |
| MIPS64 |  |  |  |  |  | 支持(2) | 支持(1) |

表中的数字表示最终选择安装的ABI的优先级，1为最高。

* 每种CPU架构都有自己对应的最优ABI。

| **CPU架构** | **最优ABI** |
| --- | --- |
| ARMv5 | armeabi |
| ARMv7 | armeabi-v7a |
| ARMv8 | arm64-v8a |
| x86 | x86 |
| x86\_64 | x86\_64 |
| MIPS | mips |
| MIPS64 | mips64 |

* 应用安装到设备时，只有该设备的CPU架构**支持的最优**so库才会被安装。

如：x86架构的设备支持x86、armeabi-v7a和armeabi等ABI。但优先级从高到低依次为x86、armeabi-v7a、armeabi。系统会根据此顺序寻找首个可用的最优的so库，找到则结束。

* **x86设备包含ARM模拟层**，能够很好地运行ARM类型的so库，但并不保证100%不发生Crash。
* **64位设备（arm64-v8a, x86\_64, mips64）能够运行32位的so库**。但是以32位模式运行时，会丢失专为64位优化过的性能特征（ART, WebView, Media, etc.）。

## NDK兼容性

* NDK平台不是向后兼容（兼容过去的版本）的，而是**向前兼容（兼容未来的版本）**的。
* NDK编译的版本应该尽量使用较低的版本，如minSdkVersion=”8”。

## Android中的so库

### so库的名称和文件名

* so库的名称可任意，如daking。
* so库的**文件名必须以lib开头**。如libdaking.so，其中lib是必要前缀，daking才是这个库的名称。

### so库文件的路径

* 在Android Studio中的路径：

默认要放在模块/src/<SourceSet>/jniLibs/<ABI>/下。如：将so库放在app/src/main/jniLibs/armeabi。

可在模块的build.gradle中修改配置指定so库的路径。如，将so库放在app/libs/armeabi下，并修改配置如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | android {  ...  sourceSets {  main {  jniLibs.srcDirs = ['libs']  }  }  }  > |

* 在Eclipse中的路径：

放在libs/<ABI>下。

* 在Android系统中的路径：

/data/data/<应用包名>/lib/下。

* 在aar包中的路径：

jni/<ABI>下。

### so库的代码加载

* **System.loadLibrary(“so库的名称”);**如System.loadLibrary(“daking”);会加载该应用下的libdaking.so文件。
* **System.load(“so库文件的全路径”);**如System.load(“/data/data/com.daking.app.demo/libdaking.so”);会加载包名为com.daking.app.demo的应用的libdaking.so文件。

### so库的使用建议

* 理论上应该为每个ABI目录提供对应的so库。但是Android支持7种ABI，若全部支持，必然导致APK包过大。
* **一般只保留armeabi、armeabi-v7a**这两个ABI的so库。

mips/mips64：极少用于手机，可忽略。

x86/x86\_64：x86架构的手机的市场占有率很低，约为1%左右。而且x86架构都包含ARM模拟层，兼容ARM类型的ABI。注意，模拟器为x86架构。

arm64-v8a：64位ARM架构。可用32位模式运行armeabi-v7a和armeabi。

armeabi-v7a：截止于2017年2月，目前主流版本AMRv7。

armeabi：老版本ARMv5，但仍需要兼容。

* 通过上面的分析，不难发现，只要提供armeabi便可兼容新/旧设备。但armeabi缺少对浮点数机器的支持，存在性能瓶颈。应该将armeabi目录中的so库拷贝一份到armeabi-v7a目录中。
* 总结：

为了减小apk体积，只保留armeabi和armeabi-v7a两个目录，并保证这两个目录中so库数量一致。

对只提供armeabi版本的第三方so库，原样复制一份到armeabi-v7a目录中。

### so库的常见错误

* 使用高版本编译出的so库运行在低版本的平台上会出错。考虑到NDK是向前兼容的，应使用低版本编译。
* so库放置的路径有误。
* 没有在每个生效的ABI目录下放置对应的so库。
  1. 若某应用有armeabi和arm64-v8a两个ABI目录，armeabi目录里有a.so和b.so，但arm64-v8a只有a.so。
  2. 当ARMv8设备在安装此应用时，根据ABI优先级，首先发现arm64-v8a目录存在，并决定使用此ABI下的so库。
  3. 但arm64-v8a目录中没有b.so，于是报错。
  4. 此时的解决方案有：一，删除arm64-v8a目录；二，arm64-v8a目录的so库情况要与armeabi一致。

# 工具

* [Native Libs Monitor](https://play.google.com/store/apps/details?id=com.xh.nativelibsmonitor.app&hl=en)，监控设备上所有已安装的App，列出它们apk中包含的so库以及检测出最终安装到的so库。

查询当前设备支持的abi

shell@kpad:/ $ getprop | grep abi

[ro.product.cpu.abi2]: [armeabi]

[ro.product.cpu.abi]: [armeabi-v7a]

[ro.product.cpu.abilist32]: [armeabi-v7a,armeabi]

[ro.product.cpu.abilist64]: []

[ro.product.cpu.abilist]: [armeabi-v7a,armeabi]

**记住一点就好：打包armeabi-v7a就好了！**

在ANE中如果SDK调用了so库，则需要把so库放到ANE下Android-ARM/lib/armeabi （调试模式）或者 armeabi-v7a（发行模式）下。

可以贴个ADT代码说明问题:

|  |
| --- |
| destApkDirectory = "lib/armeabi/";  if ((m\_configType.equals("apk")) || (hasCaptiveRuntime()))  {//发行模式 || 带运行时  destApkDirectory = "lib/armeabi-v7a/";  } |

现在还有x86的了，其实armeabi 、armeabi-v7a 和x86是编译 NDK 库时，可以使用三种支持的应用二进制接口(ABI)：  
‘armeabi’ – 默认选项，将创建以基于 ARM\* v5TE 的设备为目标的库。 具有这种目标的浮点运算使用软件浮点运算。 使用此 ABI 创建的二进制代码将可以在所有 ARM\* 设备上运行。RM的通用CPU，而v7a的。因此armeabi通用性强，但速度慢，而v7a能，在AIR打包APK调试模式adobe选择的是通用性强的armeabi模式。

‘armeabi-v7a’ – 创建支持基于 ARM\* v7 的设备的库，并将使用硬件 FPU 指令。到了android2.2：才开始支持armv7-a指令集。CPU支持硬件浮点运算，充分发挥v7a CPU的能力

‘x86’ – 生成的二进制代码可支持包含基于硬件的浮点运算的 IA-32 指令集。

对于AIR项目来说性能问题是最大的瓶颈，能优化一点就一点，所以如果接平台方SDK或者写ANE的时候使用到SO库，个人推荐全部放到armeabi-v7a下，而不要为了兼容放到armeabi。

在我们android APK的根目录有一个 libs文件夹，此文件夹下包含了armeabi 和armeabi-v7a两个文件夹

Libwebpbackport 是网络webp图片的支持，Xutil的包也包含了

## REF

[Android中的ABI](http://daking.tech/2017/08/01/Android%E4%B8%AD%E7%9A%84ABI/)

[android的armeabi和armeabi-v7a](http://www.jianshu.com/p/aed2e7898fe6)

[关于Android的.so文件你所需要知道](http://www.jianshu.com/p/cb05698a1968)的

[ABI 管理](https://developer.android.com/ndk/guides/abis.html?hl=zh-cn)

# 环境安装

## JDK

在linux上使用yum安装是非常粗暴无脑的，但仍然有需要注意的点，不然会掉坑里。这里说一下步骤。

1.执行命令yum -y list java\*查看可安装java版本。执行成功后可以看见如下的结果



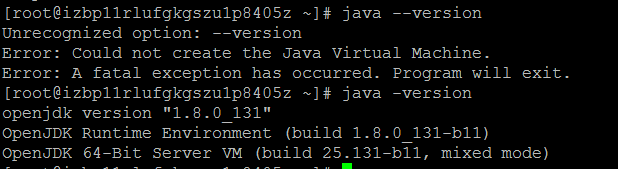
2.选择一个java版本进行安装，这里我们希望安装java1.8，因为我们的机器是64位的，所以选择安装java-1.8.0-openjdk-devel.x86\_64。

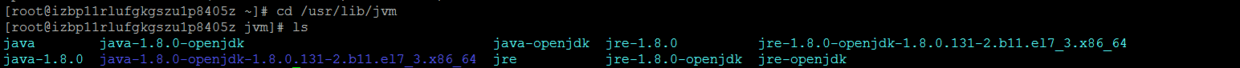
**这里有个地方要注意，上图中我用红框圈起来的两个java版本，要选择-devel的安装，因为这个安装的是jdk，而那个不带-devel的安装完了其实是jre。**

3.执行命令yum install -y java-1.8.0-openjdk-devel.x86\_64。执行完后会看见控制台刷出很多输出。

耐心等待至自动安装完成

输入java --version查看已安装的jdk版本，当出现如下输出表示安装成功。

 5.你可能好奇，yum安装的jdk，被安装到哪里去了？你可以在/usr/lib/jvm目录下找到他们。



Paste\_Image.png

至此，yum安装jdk完成。

### 三、手动安装jdk

如果你不喜欢yum安装的方式，这个很多目录不好控制，想要使用官方提供的安装包进行传统方式的安装，可以使用如下步骤。

#### .tar.gz安装

tar zxvf jdk-8u151-linux-x64.tar.gz

[root@yuanchuangke jdk8]# mv jdk1.8.0\_151 jdk8

[root@yuanchuangke lib]# mkdir kjvm

[root@yuanchuangke kjvm]# mv /home/keyguan/jdk8 .

[root@yuanchuangke etc]# vi /etc/profile

•在profile文件末尾加入：

export JAVA\_HOME=/usr/share/jdk1.6.0\_14

export PATH=$JAVA\_HOME/bin:$PATH

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

保存后执行立即生效

. /etc/profile  或者　source /etc/profile

无法更新java的版本

[root@yuanchuangke usr]# java -version

java version "1.7.0\_21"

Java(TM) SE Runtime Environment (build 1.7.0\_21-b11)

Java HotSpot(TM) 64-Bit Server VM (build 23.21-b01, mixed mode)

[root@yuanchuangke usr]# echo $JAVA\_HOME

/usr/lib/kjvm/jdk8

#### ·用文本编辑器打开/etc/profile   rpm包安装

去官网下载对应的rpm格式jdk-8u151-linux-x64.rpm为例

1.执行命令useradd java，新建用户java

Paste\_Image.png

2.执行命令passwd java，设置java用户密码

Paste\_Image.png

3.进入oracle官网，java8下载页面http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html。

Paste\_Image.png

4.选择Accept License Agreement，点击jdk-8u131-linux-x64.tar.gz，获取到下载链接。**注意，获取到的下载链接有时效（具体有效多久不清楚，反正隔天肯定不能用），请尽快复制到linux上进行下载。**

5.进入到/home/java目录下，输入wget +地址，即可开始下载

Paste\_Image.png

6.等待下载成功

Paste\_Image.png

7.查看文件，发现下载后的文件名有奇怪的后缀，重命名下载文件

Paste\_Image.png

8.输入命令tar zxvf jdk-8u131-linux-x64.tar.gz解压安装包

Paste\_Image.png

9.输入命令vim /etc/profile，打开环境变量配置文件

在文件底部输入以下信息，并保存

JAVA\_HOME=/home/java/jdk1.8.0\_131

JRE\_HOME=$JAVA\_HOME/jre

PATH=$PATH:$JAVA\_HOME/bin

CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export JAVA\_HOME

export JRE\_HOME

export PATH

export CLASSPATH

Paste\_Image.png

10.输入命令source /etc/profile，刷新环境变量配置文件使其立刻生效；输入java --version查看已安装的jdk版本

Paste\_Image.png

**你要以为这就完成了，那就掉坑里了。虽然大部分时候这就够了，但还有一步操作最好做一下。建一个/usr/bin/java的java的超链接。**

ln -s /home/java/jdk1.8.0\_131/bin/java /usr/bin/java

**为什么要建这个超链接，因为一些自己注册的linux服务（如springboot的jar注册的服务），默认情况下从/usr/bin/java路径使用java，yum安装的时候，这个超链接会自动创建，如果你自己下载包安装的话，这个超链接就需要你手动创建了。**

至此，从官网下载包安装jdk完成。

作者：低调的微胖  
链接：http://www.jianshu.com/p/10949f44ce9c  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# GitLab

## 一、系统环境

* 服务器：阿里云主机
* 操作系统：Centos7.0 64位
* 已装软件：Nginx（80端口）、Apache（8080端口）、PHP-FPM（9000端口）

## 二、安装版本

* GitLab分为社区版（GitLab Community Edition）和企业版（GitLab Enterprise Edition）。社区版免费，企业版收费，但是功能比社区版多。根据目前的需求，选择安装社区版（GitLab-CE）。
* 版本号：8.5.4

## 三、安装方式

以前试过源码安装，过程痛苦无比。此次选择官方提供的GitLab-CE Omnibus安装包。GitLab官网上有详细的安装说明，根据自己的操作系统选择相应的版本，按步骤操作即可。  
<https://about.gitlab.com/downloads>

## 四、安装过程

由于国内阿里云主机无法连接国外的GitLab Yum源，所以只能从GitLab中文社区直接下载rpm包进行安装。

curl -LJO https://mirror.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-8.5.4-ce.0.el7.x86\_64.rpm

rpm -i gitlab-ce-8.5.4-ce.0.el7.x86\_64.rpm

GitLab中文社区：<http://www.gitlab.cc>

## 五、GitLab服务构成

GitLab由以下服务构成：

* nginx：静态Web服务器
* gitlab-shell：用于处理Git命令和修改authorized keys列表
* gitlab-workhorse:轻量级的反向代理服务器
* logrotate：日志文件管理工具
* postgresql：数据库
* redis：缓存数据库
* sidekiq：用于在后台执行队列任务（异步执行）
* unicorn：An HTTP server for Rack applications，GitLab Rails应用是托管在这个服务器上面的。

重点讲一下gitlab-shell和gitlab-workhorse。

#### Gitlab Shell

GitLab Shell有两个作用：为GitLab处理Git命令、修改authorized keys列表。

当通过SSH访问GitLab Server时，GitLab Shell会：

1. 限制执行预定义好的Git命令（git push, git pull, git annex）
2. 调用GitLab Rails API 检查权限
3. 执行pre-receive钩子（在GitLab企业版中叫做Git钩子）
4. 执行你请求的动作
5. 处理GitLab的post-receive动作
6. 处理自定义的post-receive动作

当通过http(s)访问GitLab Server时，工作流程取决于你是从Git仓库拉取(pull)代码还是向git仓库推送(push)代码。如果你是从Git仓库拉取(pull)代码，GitLab Rails应用会全权负责处理用户鉴权和执行Git命令的工作；如果你是向Git仓库推送(push)代码，GitLab Rails应用既不会进行用户鉴权也不会执行Git命令，它会把以下工作交由GitLab Shell进行处理：

1. 调用GitLab Rails API 检查权限
2. 执行pre-receive钩子（在GitLab企业版中叫做Git钩子
3. 执行你请求的动作
4. 处理GitLab的post-receive动作
5. 处理自定义的post-receive动作

也许你会奇怪在通过http(s)推送(push)代码的情况下，GitLab Rails应用为什么不在GitLab Shell之前进行鉴权。这是因为GitLab Rails应用没有解析git push命令的逻辑。好的方法是将这些解析代码放在一个地方，这个地方就是GitLab Shell，这样我们就可以在通过SSH进行访问时重用这段代码。实际上，GitLabShell在执行git push命令时根本不会进行权限检查，它是依赖于pre-receive钩子进行权限检查的。而当你执行git pull命令时，权限检查是在命令执行之前的。对git pull命令的权限检查要简单得多，因为你只需要检查一个用户是否可以访问这个仓库就可以了（不需要检查分支权限）。

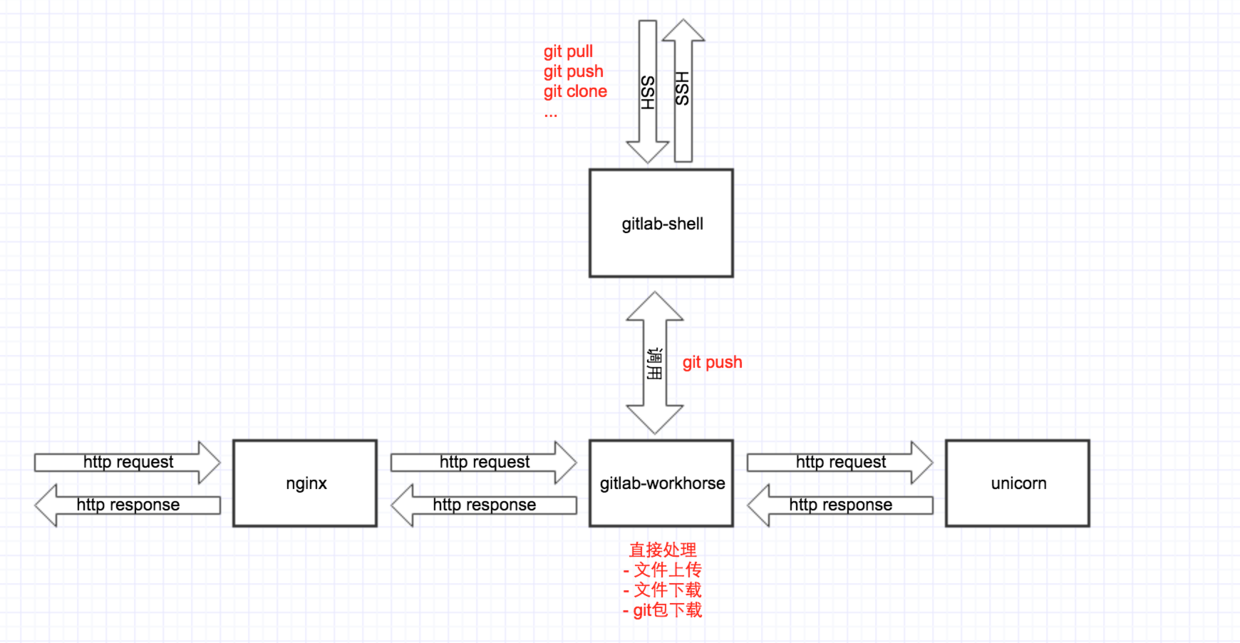
好吧，GitLab Shell这段话都是翻译官网的。链接在这里  
<https://gitlab.com/gitlab-org/gitlab-shell/blob/master/README.md>

最后一段话有点拗口，我对此还是有一点问题的：既然你把git push的逻辑都放在GitLab Shell里面了，为什么不把git pull的逻辑也都放在里面提供重用呢？  
猜想：git pull这段逻辑无法重用，因为通过http(s)方式访问时，要读取仓库的数据并且把这些数据封装成http包返回给客户端；而通过ssh方式访问时，仓库代码数据是通过ssh数据包返回的。两种访问方式返回数据的封装方式不一样，所以也没有必要提供重用。但是我觉得读取仓库数据这段逻辑应该还是重用了的。

#### GitLab Workhorse

GitLab Workhorse是一个敏捷的反向代理。它会处理一些大的HTTP请求，比如文件上传、文件下载、Git push/pull和Git包下载。其它请求会反向代理到GitLab Rails应用，即反向代理给后端的unicorn。官网对GitLab Workhorse的介绍在这里：<https://gitlab.com/gitlab-org/gitlab-workhorse/>

## 六、GitLab工作流程



GitLab工作流程图

## 七、配置

#### 配置考量

* 要求能通过子域名git.zn2studio.com访问GitLab站点并且站点内的仓库地址也要用子域名显示。
* 要求使用腾讯企业邮箱的SMTP服务器发送邮件。
* 要求使用HTTP请求方式。
* 要求能使用SSH连接方式。
* 要求避免与已装软件的端口冲突
* 要求使用系统已安装的Nginx服务器

#### 配置过程

* 修改GitLab配置文件，停用GitLab内置Nginx  
  nginx['enable'] = false
* 使用系统已经安装的Nginx给gitlab-workhorse作反向代理
* 因为unicorn的默认端口是8080,与系统已存在的Apache端口冲突，修改Apache端口为8000（也可以修改unicorn的端口）
* 修改GitLab配置文件中的external\_url  
  external\_url 'http://git.zn2studio.com'  
  修改这个配置会影响GitLab里面显示的仓库链接
* 修改GitLab邮件服务配置，使用腾讯企业邮箱的SMTP服务器
* gitlab\_rails['smtp\_enable'] = true
* gitlab\_rails['smtp\_address'] = "smtp.exmail.qq.com"
* gitlab\_rails['smtp\_port'] = 25
* gitlab\_rails['smtp\_user\_name'] = "xxx"
* gitlab\_rails['smtp\_password'] = "xxx"
* gitlab\_rails['smtp\_domain'] = "smtp.qq.com"
* gitlab\_rails['smtp\_authentication'] = 'plain'

gitlab\_rails['smtp\_enable\_starttls\_auto'] = true

## 参考

[GitLab搭建以及配置](http://www.jianshu.com/p/a22eaa1fcfe7)

# Jenkins

Jenkins是用java编写的开源持续集成工具，目前被国内外各公司广泛使用。本章教大家如何在linux服务器上安装Jenkins。

## 安装环境

CentOS7.3 64

CPU:1.2K\*40

内存：64g，目前还剩3%。。。

lscpu

free –m

软件：JDK1.8及以上

## 安装过程

Tomcat：修改端口号（<http://blog.csdn.net/jay_1989/article/details/52870760>）

从Jenkins官方网站<https://jenkins.io/>下载最新的war包

sudo wget <http://mirrors.jenkins.io/war/latest/jenkins.war>

## 配置

### 系统设置

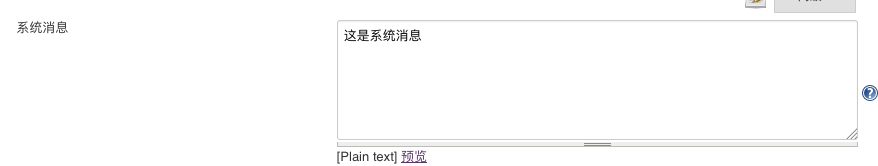
#### 主目录

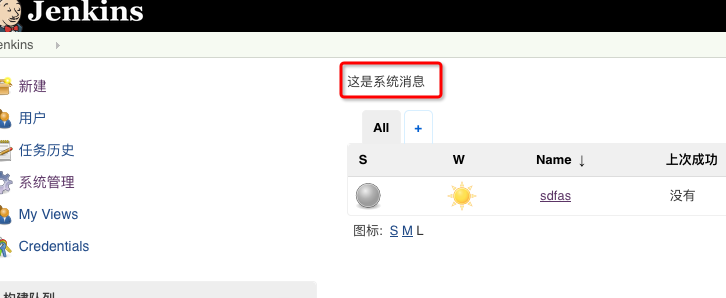
主目录是存放Jenkins所有的文件的，工作空间根目录和构建记录目录默认都是在Jenkins主目录下，这个设置一般不用进行变更

主目录，点击高级

#### 系统消息

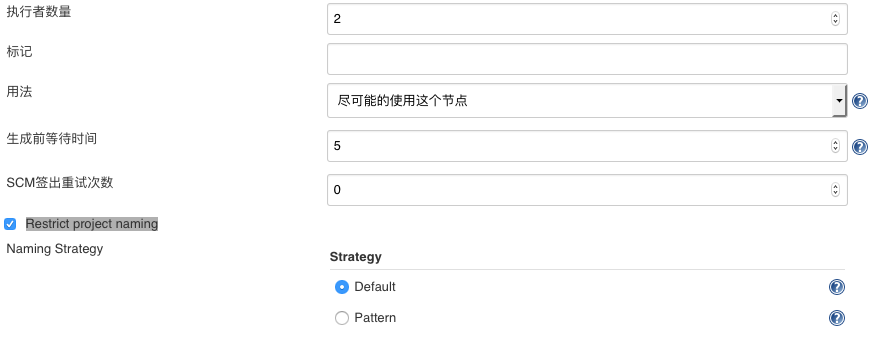
比如写：deadline之类的





#### **3.**执行者数量：

可以并发构建的数量  
 标记：用来记录这个机器的名称（为了分配节点使用，后面子节点会详细介绍）  
 用法：设置这个节点的执行策略（包括尽可能使用这个节点和只允许绑定到这台机器的job  
 生成前等待时间：这个时间为构建开始前的等待时间  
 scm签出重试次数：使用svn或者git拉取代码失败重试的次数  
Restrict project naming：限制项目命名，勾选后可以看到具体设置，可以设置为默认或者使用正则表达式进行限制

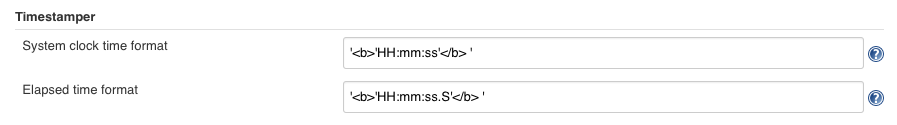


#### 4.全局属性

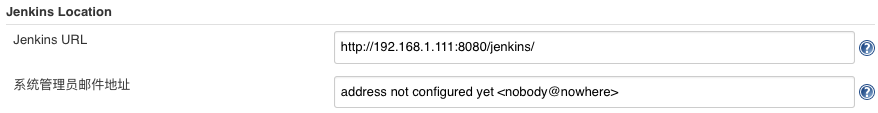
environment variables:设置全局变量，在这里定义的全局变量可以在构建或者发送邮件时引用  
tool locations:设置全局工具，可以把需要的工具都在这里进行配置，比如maven，ant,jdk等  
Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project：这就是帮助jenkinsXXXXX



5.设置时间格式



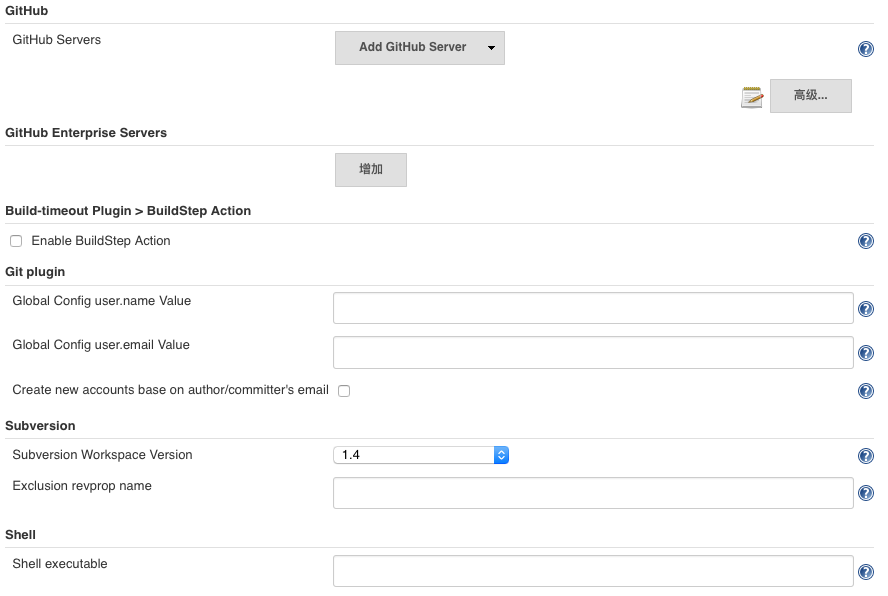
6.Jenkins location  
 jenkins url :设置jenkins的url(发送邮件引用jenkins的地址会取这个值，如果设错了，邮件的连接就会打不开)  
 系统管理员邮件地址：管理员的邮件地址（在构建需要发送邮件时，会用到这个邮件地址）



7.ssh server:设置ssh连接时的端口（如果有需要用到ssh的连接的话，就不要选择禁用了，建议指定端口或者随机选取）



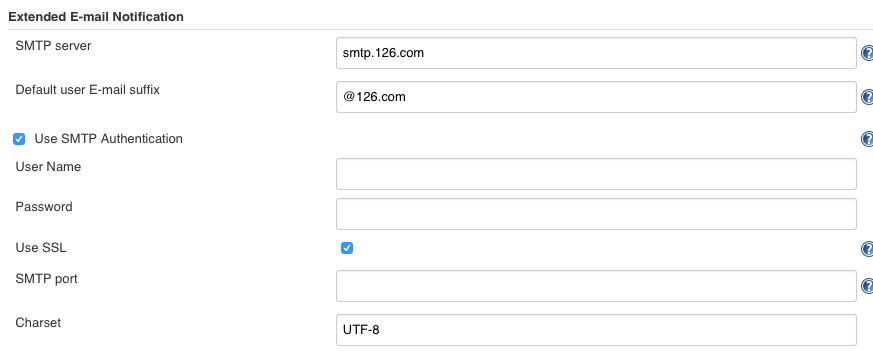
8.代码托管仓库设置：这里根据自己的配置进行设置就行了



9.shell:制定用来执行shell的执行文件，如果系统有更改可以设置，一般是不需要设置的



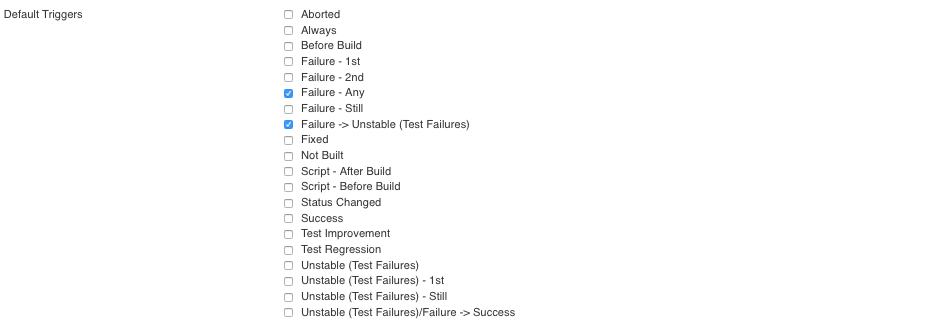
10.extended e-mail notification(邮件插件)  
SMTP server :发送邮件的服务器  
Default user E-mail suffix:默认邮箱后缀（如果配置了这个后面的邮箱就不用写后缀了）  
use smtp authentication:使用smtp用户认证，这个选项需要配合邮箱开启smtp服务，这样邮箱才能通过第三方工具发送邮件，以126的邮箱为例，[点击此链接查看开启smtp](http://jingyan.baidu.com/article/3f16e003e327772591c1039f.html?st=2&os=0&bd_page_type=1&net_type=2)  
如果开启了smtp认证  
user name:认证的邮箱  
password:认证的密码（并非邮箱的密码，是开启smtp时给的一串字符）  
如果邮箱是ssl链接，就需要勾选use ssl  
smtp port :如果你的smtp服务不是465的端口，需要配置对应的端口  
charset:邮件编码设置



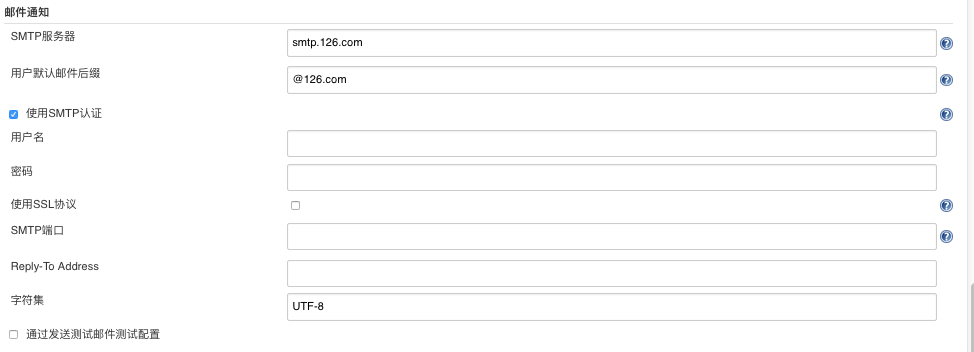
下面是设置邮件内容和发送人的一些默认设置  
default content type:设置邮件发送的格式：文本格式或者html格式  
Use List-ID Email Header：设置邮件的发送的名称（便于过滤）  
Default Recipients：设置默认的收件人  
Reply To List：设置默认回复列表  
Emergency reroute:相当于一个邮件的转发（邮件先发送到这里，然后在进行进一步处理）  
Excluded Recipients：设置接收的黑名单（就是不发送给这些人）  
Default Subject：设置默认的邮件主题  
Maximum Attachment Size：这只邮件附件的最大值  
Default Content：设置邮件的默认内容（里面可以引用一些环境变量的参数，或者插件的一些变量）  
Default Pre-send Script：在发送邮件前执行的脚本



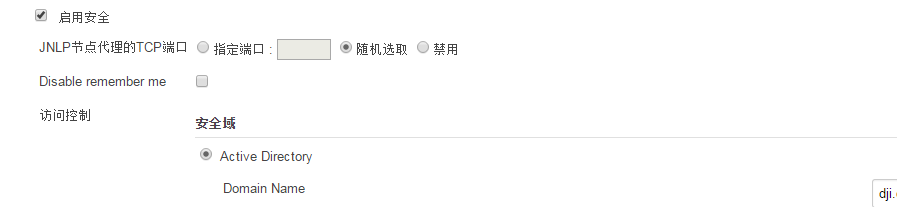
default triggers:设置默认的发送邮件策略，根据情况选择



11.邮件通知：这是默认的邮件发送工具，配置和ext email插件设置差不多，只是不能设置邮件的默认发送策略，和邮件发送的内容等信息



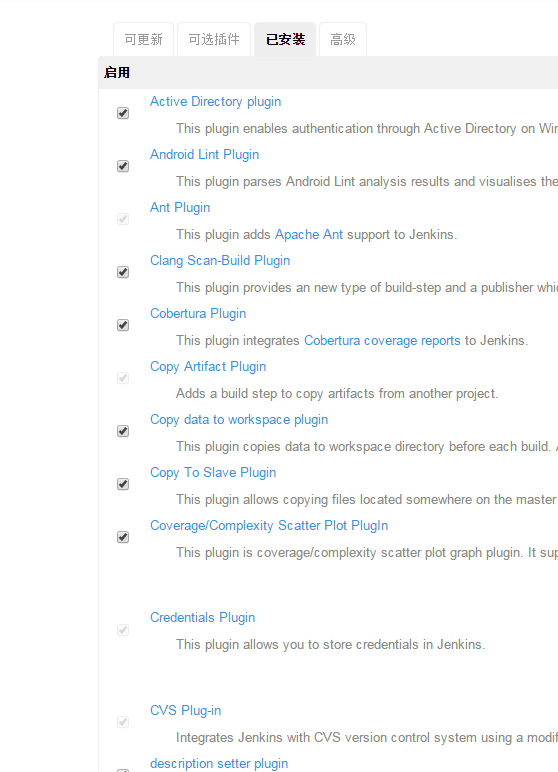
### Configure Global Security

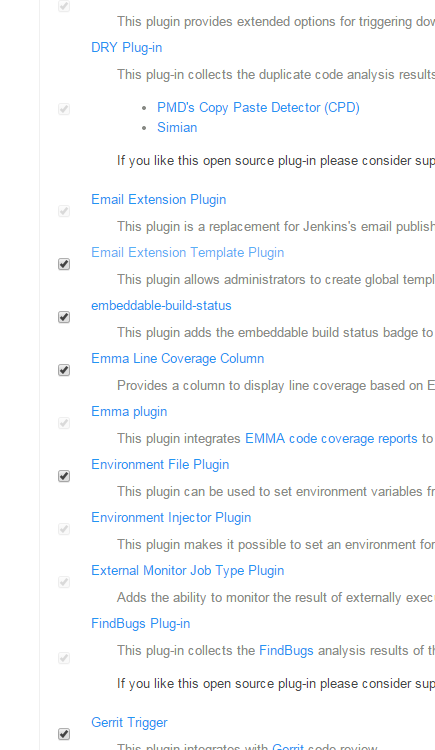


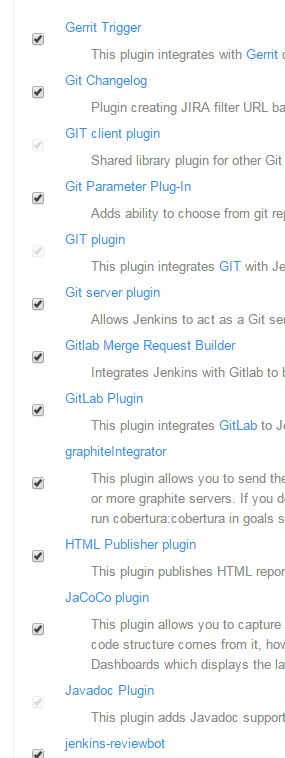


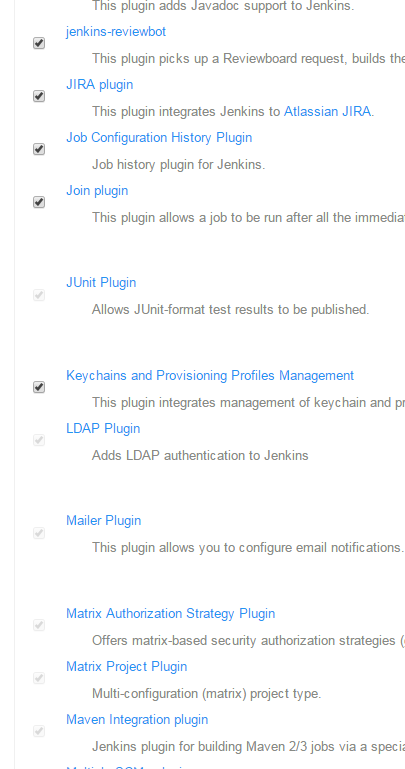
### 插件管理

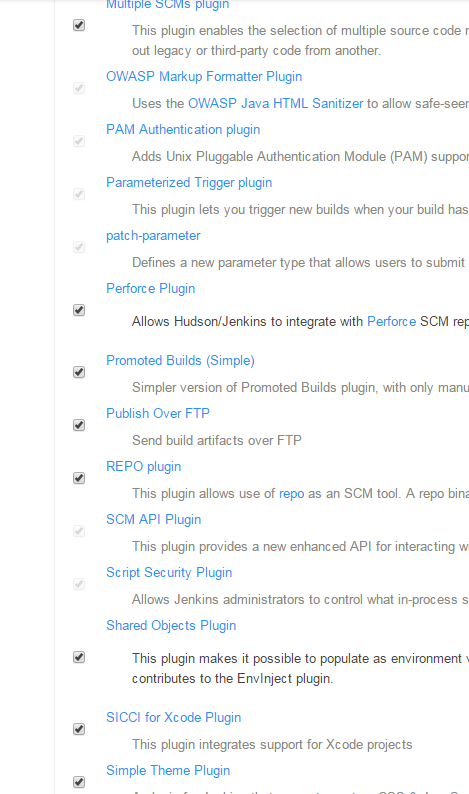
|  |  |  |
| --- | --- | --- |
| 权限认证Active Directory和LDAP | jenkins不需要重新启动 |  |
| 版本标示：Promoted Builds |  |  |

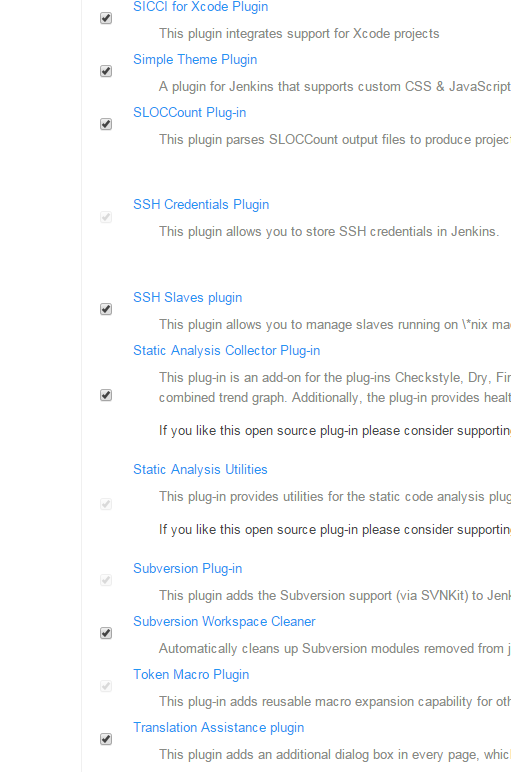


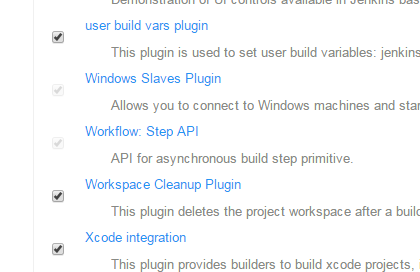












### 系统信息

# 插件

|  |  |  |  |
| --- | --- | --- | --- |
| [**名称  ↓**](http://jenkins.djicorp.com/systemInfo) | [**版本**](http://jenkins.djicorp.com/systemInfo) | [**启用**](http://jenkins.djicorp.com/systemInfo) | [**固定**](http://jenkins.djicorp.com/systemInfo) |
| active-directory | 1.41 | true | false |
| analysis-collector | 1.45 | true | false |
| analysis-core | 1.74 | true | false |
| android-lint | 2.2 | true | false |
| ant | 1.2 | true | false |
| antisamy-markup-formatter | 1.1 | true | false |
| build-user-vars-plugin | 1.5 | true | false |
| clang-scanbuild-plugin | 1.6 | true | false |
| cobertura | 1.9.7 | true | false |
| copy-data-to-workspace-plugin | 1.0 | true | false |
| copy-to-slave | 1.4.4 | true | false |
| copyartifact | 1.36.1 | true | false |
| covcomplplot | 1.1.1 | true | false |
| credentials | 1.24 | true | true |
| cvs | 2.11 | true | false |
| description-setter | 1.10 | true | false |
| downstream-ext | 1.8 | true | false |
| dry | 2.42 | true | false |
| email-ext | 2.40.5 | true | false |
| emailext-template | 0.4 | true | false |
| embeddable-build-status | 1.8 | true | false |
| emma | 1.29 | true | false |
| emmacoveragecolumn | 0.0.4 | true | false |
| envfile | 1.2 | true | false |
| envinject | 1.92.1 | true | false |
| external-monitor-job | 1.4 | true | false |
| findbugs | 4.62 | true | false |
| gerrit-trigger | 2.18.1 | true | false |
| git | 2.4.0 | true | false |
| git-changelog | 1.1 | true | false |
| git-client | 1.19.0 | true | false |
| git-parameter | 0.4.0 | true | false |
| git-server | 1.6 | true | false |
| gitlab-merge-request-jenkins | 1.2.4 | true | false |
| gitlab-plugin | 1.1.27 | true | false |
| graphiteIntegrator | 1.2 | true | false |
| htmlpublisher | 1.6 | true | false |
| jacoco | 1.0.19 | true | false |
| javadoc | 1.1 | true | false |
| jenkins-reviewbot | 2.4.6 | true | false |
| jira | 2.2.1 | true | false |
| jobConfigHistory | 2.12 | true | false |
| join | 1.16 | true | false |
| junit | 1.9 | true | true |
| kpp-management-plugin | 1.0.0 | true | false |
| ldap | 1.11 | true | false |
| mailer | 1.15 | true | true |
| matrix-auth | 1.1 | true | false |
| matrix-project | 1.4.1 | true | false |
| maven-plugin | 2.7.1 | true | false |
| multiple-scms | 0.5 | true | false |
| pam-auth | 1.1 | true | false |
| parameterized-trigger | 2.29 | true | false |
| patch-parameter | 1.2 | true | false |
| perforce | 1.3.35 | true | false |
| promoted-builds-simple | 1.9 | true | false |
| publish-over-ftp | 1.11 | true | false |
| repo | 1.8.0 | true | false |
| scm-api | 0.2 | true | false |
| script-security | 1.13 | true | false |
| shared-objects | 0.44 | true | false |
| sicci\_for\_xcode | 0.0.8 | true | false |
| simple-theme-plugin | 0.3 | true | false |
| sloccount | 1.20 | true | false |
| ssh-credentials | 1.11 | true | true |
| ssh-slaves | 1.9 | true | false |
| subversion | 1.54 | true | false |
| svn-workspace-cleaner | 1.1 | true | false |
| token-macro | 1.10 | true | false |
| translation | 1.10 | true | false |
| ui-samples-plugin | 2.0 | true | false |
| windows-slaves | 1.0 | true | false |
| workflow-step-api | 1.10 | true | false |
| ws-cleanup | 0.28 | true | false |
| xcode-plugin |  |  |  |

## 参考

[依赖JDK](http://www.jianshu.com/p/10949f44ce9c)

# Tomcat

[如何设置tomcat显示文件列表](https://jingyan.baidu.com/article/ff42efa9262df5c19e220219.html): **conf/web.xml**

**修改listings属性**为true即可。

# 文件下载S端设计

请求文件列表：

请求缩列图列表：

文件名字，文件长度，文件格式，创建时间，修改时间

下载单个文件：重定向到其他sdcard位置

断点下载：

## 参考

[Java Web端程序实现文件下载的方法分享](https://www.teakki.com/p/57df76511201d4c1629b8ec1)

[java实现文件断点续传下载功能](https://www.teakki.com/p/57df76561201d4c1629b8ef9)

# 3.文件下载C端设计

缩列图，原始图三级缓存策略

# Busybox

[android安装busybox](https://www.bbsmax.com/A/RnJW2qnrzq/)

|  |  |  |
| --- | --- | --- |
| Kw-GOOGLE |  |  |
| busybox |  |  |
| linux busybox web server-中文 | [设备产品WEB管理界面实现](https://segmentfault.com/a/1190000004963423) | 轻量级WEB Server 对比， |
|  | [编译内核+BusyBox定制一个Linux提供ssh和web服务](http://chenpipi.blog.51cto.com/8563610/1390874) | 定制一个linux系统；能让其远程登录和提供web服务 |
| android busybox web server | [android busybox httpd搭建简单web服务器](http://www.cnblogs.com/flycn/p/3677835.html) | busybox httpd -p 8080 -h /sdcard/www  关闭:*busybox ps | grep httpd*  6573 httpd  *kill 6573* |
| Android busybox web 服务器 |  |  |

## 命令学习

[Busybos文件文件命令](http://blog.csdn.net/jeffreyst_zb/article/details/7160984)

### find . –mtime

[Find–atime –ctime –mtime的用法与区别总结](http://blog.csdn.net/abcdef0966/article/details/7607545)

ind . –mtime n:  File waslast modified n\*24 hours ago.

最后一次修改发生在距离当前时间n\*24小时至(n+1)\*24 小时

find . –mtime +n:

最后一次修改发生在n+1天以前，距离当前时间为(n+1)\*24小时或者更早

find . –mtime –n:

最后一次修改发生在n天以内，距离当前时间为n\*24小时以内

### 从其他文件目录拷贝出来？

总部可能一个文件

## 有用的命令

不处理缩略图吧。

查找png文件：find /sdcard/ -name '\*.png'

busybox find /sdcard/ -name '\*.png' -and -mtime +1

## 3.3 busybox httpd

# 其他方案

[它以NGINX作为Web Server](https://wenku.baidu.com/view/a278e4c8aa00b52acfc7cac2.html?from=search)：类似于其他集成开发环境的名称，我们简单的称之为ANMPP，即Android Nginx Mysql Php-fpm Postgresql的缩写，安卓ANMPP集成环境提供了完善的PHP和NGINX模块支持。

ANMPP是需要ROOT的

[i-jetty](https://wenku.baidu.com/view/a83565e32cc58bd63186bd87.html?re=view)

[教你把安卓手机改造成web服务器,支持外网访问](https://wenku.baidu.com/view/7dd8420602768e9950e7384f.html)：[这里也有案例](https://jingyan.baidu.com/article/90bc8fc865c795f652640c59.html)

Baidu

关键字：android web服务器

1. [AndServer，一个Android端的web服务器](http://blog.csdn.net/yanzhenjie1003/article/details/64090436)：580kb，是一个android端的Web服务器，类似Apache或者Tomcat，但又有不同，它是一个普通的Android Library，[开源的](https://github.com/yanzhenjie/AndServer)
2. [NanoHttpd](http://blog.csdn.net/u012455330/article/details/48676835)
3. [如果只是一般的web service应用不依赖servlet，其实真心不必tomcat或者jetty。因为android已经带了org.apache的包，只需要简单的使用httpentity的类就可以实现简单的web server](https://zhidao.baidu.com/question/555463599.html)

Guge:

关键字：

# [android安装busybox](https://www.bbsmax.com/A/RnJW2qnrzq/)

0. 把手机给Root。

1. [下载BusyBox的binary](http://www.busybox.net/downloads/binaries) ，选择最新版本，然后下载对应你的设备架构的版本 （ /proc/cpuinfo），这里我下载了busybox-armv6l，重命名为busybox，下面将以这个文件名为示例。

# NanoHTTPD

KW: NanoHTTPD

[NanoHttpd嵌入式服务器](http://www.jianshu.com/p/1d53ceb6b4ac)

[**基于NanoHttpd的Android视频服务器开发**](http://ticktick.blog.51cto.com/823160/1705767)

[**源码**](https://github.com/NanoHttpd/nanohttpd)**,**

[**源码分析cn**](http://blog.csdn.net/new_abc/article/details/52876681)**1，**

**[源码分析cn2](http://www.it165.net/pro/html/201402/9405.html)**

[Simple webserver](http://www.voidcn.com/blog/new_abc/article/p-6241847.html)

[该是浏览器多请求了一个 /favicon.ico](http://www.jianshu.com/p/1d53ceb6b4ac)

# 跨进程通信

## 传输大数据传输

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| RPC/AIDL文件 | Binder进程间通信，需要内核中拷贝一次内存 | 声明权限保护好 |  |
| LocalSocket | 如果想要Java和C之间更灵活的通信方式，可以使用，只需要Internet权限，实质是Unix Domain Socket（不是通常的network socket，虽然二者接口基本一致），需要内核中拷贝两次内存 | 需要在两端做好校验 |  |
| 共享内存 | 最高效的方法   1. Android里边有个Ashmem, 不需要内存拷贝，需要同步，为何有人说不适合 2. android有匿名共享内存，用MemoryFile | 系统级的权限。 |  |

搜索：Android 共享内存，

## 参考

[在Android中两个进程之间传输大数据](https://www.zhihu.com/question/51478014)

# Linux工具

## RPM

在 Linux 操作系统下，几乎所有的软件均通过RPM 进行安装、卸载及管理等操作。RPM 的全称为Redhat Package Manager ，是由Redhat 公司提出的，用于管理Linux 下软件包的软件。Linux 安装时，除了几个核心模块以外，其余几乎所有的模块均通过RPM 完成安装。RPM 有五种操作模式，分别为：安装、卸载、升级、查询和验证。

PM 安装操作

命令：

rpm -i 需要安装的包文件名

举例如下：

rpm -i example.rpm 安装 example.rpm 包；

rpm -iv example.rpm 安装 example.rpm 包并在安装过程中显示正在安装的文件信息；

rpm -ivh example.rpm 安装 example.rpm 包并在安装过程中显示正在安装的文件信息及安装进度；

RPM 查询操作

命令：

rpm -q …

附加查询命令：

a 查询所有已经安装的包以下两个附加命令用于查询安装包的信息；

i 显示安装包的信息；

l 显示安装包中的所有文件被安装到哪些目录下；

s 显示安装版中的所有文件状态及被安装到哪些目录下；以下两个附加命令用于指定需要查询的是安装包还是已安装后的文件；

p 查询的是安装包的信息；

f 查询的是已安装的某文件信息；

举例如下：

rpm -qa | grep tomcat4 查看 tomcat4 是否被安装；

rpm -qip example.rpm 查看 example.rpm 安装包的信息；

rpm -qif /bin/df 查看/bin/df 文件所在安装包的信息；

rpm -qlf /bin/df 查看/bin/df 文件所在安装包中的各个文件分别被安装到哪个目录下；

RPM 卸载操作

命令：

rpm -e 需要卸载的安装包

在卸载之前，通常需要使用rpm -q …命令查出需要卸载的安装包名称。

举例如下：

rpm -e tomcat4 卸载 tomcat4 软件包

RPM 升级操作

命令：

rpm -U 需要升级的包

举例如下：

rpm -Uvh example.rpm 升级 example.rpm 软件包

RPM 验证操作

命令：

rpm -V 需要验证的包

举例如下：

rpm -Vf /etc/tomcat4/tomcat4.conf

输出信息类似如下：

S.5....T c /etc/tomcat4/tomcat4.conf

其中，S 表示文件大小修改过，T 表示文件日期修改过。限于篇幅，更多的验证信息请您参考rpm 帮助文件：man rpm

RPM 的其他附加命令

--force 强制操作 如强制安装删除等；

--requires 显示该包的依赖关系；

--nodeps 忽略依赖关系并继续操作；

.

## .tar.gz

.tar.gz 和 .tgz

解压：tar zxvf FileName.tar.gz

压缩：tar zcvf FileName.tar.gz DirName

# Shell脚本

Alpha：是内部测试版,一般不向外部发布,会有很多Bug.一般只有测试人员使用。  
  
Beta：也是测试版，这个阶段的版本会一直加入新的功能。在Alpha版之后推出。  
  
RC：(Release　Candidate) 顾名思义么 ! 用在软件上就是候选版本。系统平台上就是发行候选版本。RC版不会再加入新的功能了，主要着重于除错。  
  
GA:General Availability,正式发布的版本，在国外都是用GA来说明release版本的。  
  
RTM：(Release to Manufacture)是给工厂大量压片的版本，内容跟正式版是一样的，不过RTM版也有出限制、评估版的。但是和正式版本的主要程序代码都是一样的。  
  
OEM：是给计算机厂商随着计算机贩卖的，也就是随机版。只能随机器出货，不能零售。只能全新安装，不能从旧有操作系统升级。包装不像零售版精美，通常只有一面CD和说明书(授权书)。   
  
RVL：号称是正式版，其实RVL根本不是版本的名称。它是中文版/英文版文档破解出来的。   
  
EVAL：而流通在网络上的EVAL版，与“评估版”类似，功能上和零售版没有区别。   
  
RTL：Retail(零售版)是真正的正式版，正式上架零售版。在安装盘的i386文件夹里有一个eula.txt，最后有一行EULAID，就是你的版本。比如简体中文正式版是EULAID:WX.4\_PRO\_RTL\_CN，繁体中文正式版是WX.4\_PRO\_RTL\_TW。其中：如果是WX.开头是正式版，WB.开头是测试版。\_PRE，代表家庭版；\_PRO，代表专业版。  
  
α、β、λ常用来表示[软件测试](javascript:;)过程中的三个阶段，α是第一阶段，一般只供内部测试使用；β是第二个阶段，已经消除了软件中大部分的不完善之处，但仍有可能还存在缺陷和漏洞，一般只提供给特定的用户群来测试使用；λ是第三个阶段，此时产品已经相当成熟，只需在个别地方再做进一步的优化处理即可上市发行。

# Learn

Maven

Git+Jenkins之自动构建