linux 下编辑 rc.local 设置开机启动

设置开机启动示例：

```
# touch /etc/init.d/rc.local
```

设置为可执行：

```
# chmod +x /etc/init.d/rc.local
```

用 update-rc.d 设置启动级别：

```
# update-rc.d rc.local start 99 2 3 4 5 . stop 99 0 1 6 .
```

为了编辑方便，创建一个链接：

```
# ln -s /etc/init.d/rc.local /etc/rc.local
# cat /etc/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
## start apache
/usr/local/apache/bin/apachectl start
```

以下时对与 rc.local 的详细讲述：

linux 有自己一套完整的启动体系，抓住了 linux 启动的脉络，linux 的启动过程将不再神秘。

本文中假设 inittab 中设置的 init tree 为：

```
/etc/rc.d/rc0.d
/etc/rc.d/rc1.d
/etc/rc.d/rc2.d
/etc/rc.d/rc3.d
/etc/rc.d/rc4.d
/etc/rc.d/rc5.d
/etc/rc.d/rc6.d
/etc/rc.d/init.d
```

目录

1. 关于 linux 的启动

init 是所有进程的顶层

init 读取/etc/inittab，执行 rc.sysinit 脚本

(注意文件名是不一定的,有些 unix 甚至会将语句直接写在 inittab 中)

rc.sysinit 脚本作了很多工作:

```
init $PATH
config network
start swap function
set hostname
check root file system, repair if needed
check root space
....
```

rc.sysinit 根据 inittab 执行 rc?.d 脚本
linux 是多用户系统，getty 是多用户与单用户的分水岭
在 getty 之前运行的是系统脚本

2. 关于 rc.d
所有启动脚本放置在 /etc/rc.d/init.d 下
rc?.d 中放置的是 init.d 中脚本的链接，命名格式是:
S{number}{name}
K{number}{name}
S 开始的文件向脚本传递 start 参数
K 开始的文件向脚本传递 stop 参数
number 决定执行的顺序

3. 启动脚本示例
这是一个用来启动 httpd 的 /etc/rc.d/init.d/apache 脚本：
代码:

```
#!/bin/bash
.....
```

可以看出他接受 start,stop,restart,status 参数
然后可以这样建立 rc?.d 的链接：
代码:

```
cd /etc/rc.d/init.d &&
ln -sf ../init.d/apache ../rc0.d/K28apache &&
ln -sf ../init.d/apache ../rc1.d/K28apache &&
ln -sf ../init.d/apache ../rc2.d/K28apache &&
ln -sf ../init.d/apache ../rc3.d/S32apache &&
ln -sf ../init.d/apache ../rc4.d/S32apache &&
ln -sf ../init.d/apache ../rc5.d/S32apache &&
ln -sf ../init.d/apache ../rc6.d/K28apache
```

4. 关于 rc.local
经常使用的 rc.local 则完全是习惯问题，不是标准。
各个发行版有不同的实现方法，可以这样实现：
代码:

```
touch /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc1.d/S999rc.local &&
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc2.d/S999rc.local &&
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc3.d/S999rc.local &&
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc4.d/S999rc.local &&
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc5.d/S999rc.local &&
ln -sf /etc/rc.d/rc.local /etc/rc.d/rc6.d/S999rc.local
```

5. 关于 bash 启动脚本
/etc/profile
/etc/bashrc
~/.bash_profile
~/.bashrc
是 bash 的启动脚本

一般用来设置单用户的启动环境，也可以实现开机单用户的程序，但要明确他们都是属于 bash 范畴而不是系统范畴。
他们的具体作用介绍如下：
/bin/bash 这个命令解释程序(后面简称 shell)使用了一系列启动文件来建立一个运行环境：
/etc/profile
/etc/bashrc
~/.bash_profile
~/.bashrc
~/.bash_logout
每一个文件都有特殊的功用并对登陆和交互环境有不同的影响。
/etc/profile 和 ~/.bash_profile 是在启动一个交互登陆 shell 的时候被调用。
/etc/bashrc 和 ~/.bashrc 是在一个交互的非登陆 shell 启动的时候被调用。
~/.bash_logout 在用户注销登陆的时候被读取
一个交互的登陆 shell 会在 /bin/login 成功登陆之后运行。一个交互的非登陆 shell 是通过命令行来运行的，如[prompt]$/bin/bash。一般一个非交互的 shell 出现在运行 shell 脚本的时候。之所以叫非交互的 shell，是因为它不在命令行上等待输入而只是执行脚本程序。
6. 关于开机程序的自动启动
系统脚本可以放置在/etc/rc.d/init.d 中并建立/etc/rc.d/rc?.d 链接，也可以直接放置在/etc/rc.d/rc.local 中。
init.d 脚本包含完整的 start,stop,status,reload 等参数，是标准做法，推荐使用。
为特定用户使用的程序（如有的用户需要使用中文输入法而有的不需要）放置在~/中的 bash 启动脚本中。
==========
设置系统自动启动
在/etc/init.d/下创建 smsafe 文件
内容：

```
#!/bin/bash
# chkconfig: 35 95 1
# description: script to start/stop smsafe
case $1 in
start)
sh /opt/startsms.sh
;;
stop)
sh /opt/stopsms.sh
;;
*)
echo "Usage: $0 (start|stop)"
;;
esac
```

更改权限

```
# chmod 775 smsafe
```

加入自动启动

```
# chkconfig –add smsafe
```

查看自动启动设置

```
# chkconfig –list smsafe
smsafe 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

以后可以用以下命令启动和停止脚本

```
# service smsafe start 启动
# service smsafe stop 停止
```

==========

jira 的启动主要依靠的是 bin 目录下的 catalina.sh 脚本，提供了如 init 脚本的 start，stop 等参数

```
#!/bin/bash
#
# chkconfig: 2345 85 15
# description: jira
# processname: jira
# source function library
. /etc/init.d/functions
#下面一行比较重要，为 jira 的安装路径，没有的话，将会提示找不到文件
CATALINA_HOME="/var/www/jira"
RETVAL=0
start() {
echo -n $"Starting jira services: "
. /var/www/jira/bin/catalina.sh start
RETVAL=$?
echo
}
stop() {
echo -n $"Shutting down jira services: "
. /var/www/jira/bin/catalina.sh stop
RETVAL=$?
echo
}
case "$1" in
start)
start
;;
stop)
stop
;;
restart|reload)
stop
start
;;
status)
status jira
RETVAL=$?
;;
*)
echo $"Usage: $0 {start|stop|restart|status}"
exit 1
esac
exit $RETVAL
```

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

保存为/etc/init.d/jira

然后利用 chkconfig --add jira

OK

启动/etc/init.d/jira start

停止/etc/init.d/jira stop

==========
(以 Websphere 为例子)
1. 在/etc/rc.d/init.d 目录下新建启动脚本 startWebsphere，键入以下内容：
#!/bin/sh
/opt/WebSphere/AppServer/bin/startServer.sh server1
修改该文件的权限：
chmod 755 startWebsphere
2. 在对应的目录下建立软连接(假设系统默认进入 X11)
cd /etc/rc.d/rc5.d
ln -s ../init.d/startWebsphere S99startWebsphere
3. 重启系统即可
==========
linux 下 oracle 的自启动脚本
1.写一个 StartOracle.sql,假设放在/目录下
vi /StartOracle.sql 加入如下两行保存
startup
exit
2.配置/etc/rc.local
vi /etc/rc.local 加入如下内容，保存
su - oracle -c '$ORACLE_HOME/bin/lsnrctl start'
su - oracle -c '$ORACLE_HOME/bin/sqlplus "/as sysdba" @/StartOracle.sql'
3. 如果还要自动启动 oracle enterprise manager(em)和 isqlplus 可以如下配置
vi /etc/rc.local 加入：
su - oracle -c '$ORACLE_HOME/bin/emctl start dbconsole'
su - oracle -c '$ORACLE_HOME/bin/isqlplusctl start'
要知道 em 和 isqlplus 等使用的端口可以查询文件：
$ORACLE_HOME/install/portlist.ini(以 oracle 10.1.0.3 为例)
==========
#root 命令行下直接绑定演示：
arp -s 192.x.x.x. 00:ea:se 绑定.
arp -d 删除
arp -f 批量导入