# 1 数组回顾

数组 Array

数组包含许多元素element。

如何定义数组？

array\_name(‘’element0” “element1” ……)

array\_name[0]=”element0”

如何查看数组

echo ${array\_name[0]}

echo $array\_name #输出显示element0的值

如何显示数组元素个数

echo ${#array\_name[\*]}

# 2 curl命令

curl [option] http//：

--cacert <file> CA证书 (SSL)

--capath <directory> CA目录 (made using c\_rehash) to verify peer against (SSL)

--compressed 要求返回是压缩的形势 (using deflate or gzip)

--connect-timeout <seconds> 设置最大请求时间

-H/--header <line>自定义头信息传递给服务器

-i/--include 输出时包括protocol头信息

-I/--head 只显示文档信息

--interface <interface> 使用指定网络接口/地址

-s/--silent静音模式。不输出任何东西

-u/--user <user[:password]>设置服务器的用户和密码

-p/--proxytunnel 使用HTTP代理

# 3 检测脚本

RS健康状态检查脚本示例第一版：

#!/bin/bash

#

VIP=192.168.10.3

CPORT=80

FAIL\_BACK=127.0.0.1

FBSTATUS=0

RS=("192.168.10.7" "192.168.10.8")

RSTATUS=("1" "1")

RW=("2" "1")

RPORT=80

TYPE=g

add() {

ipvsadm -a -t $VIP:$CPORT -r $1:$RPORT -$TYPE -w $2

[ $? -eq 0 ] && return 0 || return 1

}

del() {

ipvsadm -d -t $VIP:$CPORT -r $1:$RPORT

[ $? -eq 0 ] && return 0 || return 1

}

while :; do

let COUNT=0

for I in ${RS[\*]}; do

if curl --connect-timeout 1 http://$I &> /dev/null; then

if [ ${RSTATUS[$COUNT]} -eq 0 ]; then

add $I ${RW[$COUNT]}

[ $? -eq 0 ] && RSTATUS[$COUNT]=1

fi

else

if [ ${RSTATUS[$COUNT]} -eq 1 ]; then

del $I

[ $? -eq 0 ] && RSTATUS[$COUNT]=0

fi

fi

let COUNT++

done

sleep 5

done

RS健康状态检查脚本示例第二版：

#!/bin/bash

#

VIP=192.168.10.3

CPORT=80

FAIL\_BACK=127.0.0.1

RS=("192.168.10.7" "192.168.10.8")

declare -a RSSTATUS

RW=("2" "1")

RPORT=80

TYPE=g

CHKLOOP=3

LOG=/var/log/ipvsmonitor.log

addrs() {

ipvsadm -a -t $VIP:$CPORT -r $1:$RPORT -$TYPE -w $2

[ $? -eq 0 ] && return 0 || return 1

}

delrs() {

ipvsadm -d -t $VIP:$CPORT -r $1:$RPORT

[ $? -eq 0 ] && return 0 || return 1

}

checkrs() {

local I=1

while [ $I -le $CHKLOOP ]; do

if curl --connect-timeout 1 http://$1 &> /dev/null; then

return 0

fi

let I++

done

return 1

}

initstatus() {

local I

local COUNT=0;

for I in ${RS[\*]}; do

if ipvsadm -L -n | grep "$I:$RPORT" && > /dev/null ; then

RSSTATUS[$COUNT]=1

else

RSSTATUS[$COUNT]=0

fi

let COUNT++

done

}

initstatus

while :; do

let COUNT=0

for I in ${RS[\*]}; do

if checkrs $I; then

if [ ${RSSTATUS[$COUNT]} -eq 0 ]; then

addrs $I ${RW[$COUNT]}

[ $? -eq 0 ] && RSSTATUS[$COUNT]=1 && echo "`date +'%F %H:%M:%S'`, $I is back." >> $LOG

fi

else

if [ ${RSSTATUS[$COUNT]} -eq 1 ]; then

delrs $I

[ $? -eq 0 ] && RSSTATUS[$COUNT]=0 && echo "`date +'%F %H:%M:%S'`, $I is gone." >> $LOG

fi

fi

let COUNT++

done

sleep 5

done

# 4 整体操作

libnet下载地址： http://search.cpan.org/dist/libnet/

ipvsadm下载地址： http://www.linuxvirtualserver.org/software/ipvs.html#kernel-2.6

从Linux内核版本2.6起，ip\_vs code已经被整合进了内核中，因此，只要在编译内核的时候选择了ipvs的功能，您的Linux即能支持LVS。Linux 2.4.23以后的内核版本也整合了ip\_vs code，但如果是更旧的内核版本，您得自己手动将ip\_vs code整合进内核原码中，并重新编译内核方可使用lvs。

一、关于ipvsadm:

ipvsadm是运行于用户空间、用来与ipvs交互的命令行工具，它的作用表现在：

1、定义在Director上进行dispatching的服务(service)，以及哪此服务器(server)用来提供此服务；

2、为每台同时提供某一种服务的服务器定义其权重（即概据服务器性能确定的其承担负载的能力）；

注：权重用整数来表示，有时候也可以将其设置为atomic\_t；其有效表示值范围为24bit整数空间，即（2^24-1）；

因此，ipvsadm命令的主要作用表现在以下方面：

1、添加服务（通过设定其权重>0）；

2、关闭服务（通过设定其权重>0）；此应用场景中，已经连接的用户将可以继续使用此服务，直到其退出或超时；新的连接请求将被拒绝；

3、保存ipvs设置，通过使用“ipvsadm-sav > ipvsadm.sav”命令实现；

4、恢复ipvs设置，通过使用“ipvsadm-sav < ipvsadm.sav”命令实现；

5、显示ip\_vs的版本号，下面的命令显示ipvs的hash表的大小为4k；

# ipvsadm

IP Virtual Server version 1.2.1 (size=4096)

6、显示ipvsadm的版本号

# ipvsadm --version

ipvsadm v1.24 2003/06/07 (compiled with popt and IPVS v1.2.0)

二、ipvsadm使用中应注意的问题

默认情况下，ipvsadm在输出主机信息时使用其主机名而非IP地址，因此，Director需要使用名称解析服务。如果没有设置名称解析服务、服务不可用或设置错误，ipvsadm将会一直等到名称解析超时后才返回。当然，ipvsadm需要解析的名称仅限于RealServer，考虑到DNS提供名称解析服务效率不高的情况，建议将所有RealServer的名称解析通过/etc/hosts文件来实现；

三、调度算法

Director在接收到来自于Client的请求时，会基于"schedule"从RealServer中选择一个响应给Client。ipvs支持以下调度算法：

1、轮询（round robin, rr),加权轮询(Weighted round robin, wrr)——新的连接请求被轮流分配至各RealServer；算法的优点是其简洁性，它无需记录当前所有连接的状态，所以它是一种无状态调度。轮叫调度算法假设所有服务器处理性能均相同，不管服务器的当前连接数和响应速度。该算法相对简单，不适用于服务器组中处理性能不一的情况，而且当请求服务时间变化比较大时，轮叫调度算法容易导致服务器间的负载不平衡。

2、最少连接(least connected, lc)， 加权最少连接(weighted least connection, wlc)——新的连接请求将被分配至当前连接数最少的RealServer；最小连接调度是一种动态调度算法，它通过服务器当前所活跃的连接数来估计服务器的负载情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加1；当连接中止或超时，其连接数减一。

3、基于局部性的最少链接调度（Locality-Based Least Connections Scheduling，lblc）——针对请求报文的目标IP地址的负载均衡调度，目前主要用于Cache集群系统，因为在Cache集群中客户请求报文的目标IP地址是变化的。这里假设任何后端服务器都可以处理任一请求，算法的设计目标是在服务器的负载基本平衡情况下，将相同目标IP地址的请求调度到同一台服务器，来提高各台服务器的访问局部性和主存Cache命中率，从而整个集群系统的处理能力。LBLC调度算法先根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于其一半的工作负载，则用“最少链接”的原则选出一个可用的服务器，将请求发送到该服务器。

4、带复制的基于局部性最少链接调度（Locality-Based Least Connections with Replication Scheduling，lblcr）——也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个目标IP地址到一组服务器的映射，而 LBLC算法维护从一个目标IP地址到一台服务器的映射。对于一个“热门”站点的服务请求，一台Cache 服务器可能会忙不过来处理这些请求。这时，LBLC调度算法会从所有的Cache服务器中按“最小连接”原则选出一台Cache服务器，映射该“热门”站点到这台Cache服务器，很快这台Cache服务器也会超载，就会重复上述过程选出新的Cache服务器。这样，可能会导致该“热门”站点的映像会出现在所有的Cache服务器上，降低了Cache服务器的使用效率。LBLCR调度算法将“热门”站点映射到一组Cache服务器（服务器集合），当该“热门”站点的请求负载增加时，会增加集合里的Cache服务器，来处理不断增长的负载；当该“热门”站点的请求负载降低时，会减少集合里的Cache服务器数目。这样，该“热门”站点的映像不太可能出现在所有的Cache服务器上，从而提供Cache集群系统的使用效率。LBLCR算法先根据请求的目标IP地址找出该目标IP地址对应的服务器组；按“最小连接”原则从该服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载；则按“最小连接”原则从整个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

5、目标地址散列调度（Destination Hashing，dh）算法也是针对目标IP地址的负载均衡，但它是一种静态映射算法，通过一个散列（Hash）函数将一个目标IP地址映射到一台服务器。目标地址散列调度算法先根据请求的目标IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

6、源地址散列调度（Source Hashing，sh）算法正好与目标地址散列调度算法相反，它根据请求的源IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。它采用的散列函数与目标地址散列调度算法的相同。除了将请求的目标IP地址换成请求的源IP地址外，它的算法流程与目标地址散列调度算法的基本相似。在实际应用中，源地址散列调度和目标地址散列调度可以结合使用在防火墙集群中，它们可以保证整个系统的唯一出入口。

四、关于LVS追踪标记fwmark：

如果LVS放置于多防火墙的网络中，并且每个防火墙都用到了状态追踪的机制，那么在回应一个针对于LVS的连接请求时必须经过此请求连接进来时的防火墙，否则，这个响应的数据包将会被丢弃。

查看LVS上当前的所有连接

# ipvsadm -Lcn

或者

#cat /proc/net/ip\_vs\_conn

查看虚拟服务和RealServer上当前的连接数、数据包数和字节数的统计值，则可以使用下面的命令实现：

# ipvsadm -l --stats

查看包传递速率的近似精确值，可以使用下面的命令：

# ipvsadm -l --rate

VS/NAT

LVS-NAT基于cisco的LocalDirector。VS/NAT不需要在RealServer上做任何设置，其只要能提供一个tcp/ip的协议栈即可，甚至其无论基于什么OS。基于VS/NAT，所有的入站数据包均由Director进行目标地址转换后转发至内部的RealServer，RealServer响应的数据包再由Director转换源地址后发回客户端。

VS/NAT模式不能与netfilter兼容，因此，不能将VS/NAT模式的Director运行在netfilter的保护范围之中。现在已经有补丁可以解决此问题，但尚未被整合进ip\_vs code。

\_\_\_\_\_\_\_\_\_\_\_\_

| |

| client |

|\_\_\_\_\_\_\_\_\_\_\_\_|

CIP=192.168.0.253 (eth0)

|

|

VIP=192.168.0.220 (eth0)

\_\_\_\_\_\_\_\_\_\_\_\_

| |

| director |

|\_\_\_\_\_\_\_\_\_\_\_\_|

DIP=192.168.10.10 (eth1)

|

(switch)------------------------

| |

RIP=192.168.10.2 (eth0) RIP=192.168.10.3 (eth0)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

| | | |

| realserver1 | | realserver2 |

|\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_|

设置VS/NAT模式的LVS(这里以web服务为例)

Director:

建立服务

# ipvsadm -A -t VIP:PORT -s rr

如:

# ipvsadm -A -t 192.168.0.220:80 -s rr

设置转发：

# ipvsadm -a -t VIP:PORT -r RIP\_N:PORT -m -w N

如：

# ipvsadm -a -t 192.168.0.220:80 -r 192.168.10.2 -m -w 1

# ipvsadm -a -t 192.168.0.220:80 -r 192.168.10.3 -m -w 1

打开路由转发功能

# echo "1" > /proc/sys/net/ipv4/ip\_forward

服务控制脚本：

#!/bin/bash

#

# chkconfig: - 88 12

# description: LVS script for VS/NAT

#

. /etc/rc.d/init.d/functions

#

VIP=192.168.0.219

DIP=192.168.10.10

RIP1=192.168.10.11

RIP2=192.168.10.12

#

case "$1" in

start)

/sbin/ifconfig eth0:1 $VIP netmask 255.255.255.0 up

# Since this is the Director we must be able to forward packets

echo 1 > /proc/sys/net/ipv4/ip\_forward

# Clear all iptables rules.

/sbin/iptables -F

# Reset iptables counters.

/sbin/iptables -Z

# Clear all ipvsadm rules/services.

/sbin/ipvsadm -C

# Add an IP virtual service for VIP 192.168.0.219 port 80

# In this recipe, we will use the round-robin scheduling method.

# In production, however, you should use a weighted, dynamic scheduling method.

/sbin/ipvsadm -A -t $VIP:80 -s rr

# Now direct packets for this VIP to

# the real server IP (RIP) inside the cluster

/sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -m

/sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -m

/bin/touch /var/lock/subsys/ipvsadm.lock

;;

stop)

# Stop forwarding packets

echo 0 > /proc/sys/net/ipv4/ip\_forward

# Reset ipvsadm

/sbin/ipvsadm -C

# Bring down the VIP interface

ifconfig eth0:1 down

rm -rf /var/lock/subsys/ipvsadm.lock

;;

status)

[ -e /var/lock/subsys/ipvsadm.lock ] && echo "ipvs is running..." || echo "ipvsadm is stopped..."

;;

\*)

echo "Usage: $0 {start|stop}"

;;

esac

ARP问题：

\_\_\_\_\_\_\_\_\_\_

| |

| client |

|\_\_\_\_\_\_\_\_|

|

|

(router)

|

|

| \_\_\_\_\_\_\_\_\_\_

| DIP | |

|------| director |

| VIP |\_\_\_\_\_\_\_\_\_\_|

|

|

|

------------------------------------

| | |

| | |

RIP1, VIP RIP2, VIP RIP3, VIP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

| | | | | |

| realserver1 | | realserver2 | | realserver3 |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

在如上图的VS/DR或VS/TUN应用的一种模型中（所有机器都在同一个物理网络），所有机器（包括Director和RealServer）都使用了一个额外的IP地址，即VIP。当一个客户端向VIP发出一个连接请求时，此请求必须要连接至Director的VIP，而不能是RealServer的。因为，LVS的主要目标就是要Director负责调度这些连接请求至RealServer的。

因此，在Client发出至VIP的连接请求后，只能由Director将其MAC地址响应给客户端（也可能是直接与Director连接的路由设备），而Director则会相应的更新其ipvsadm table以追踪此连接，而后将其转发至后端的RealServer之一。

如果Client在请求建立至VIP的连接时由某RealServer响应了其请求，则Client会在其MAC table中建立起一个VIP至RealServer的对就关系，并以至进行后面的通信。此时，在Client看来只有一个RealServer而无法意识到其它服务器的存在。

为了解决此问题，可以通过在路由器上设置其转发规则来实现。当然，如果没有权限访问路由器并做出相应的设置，则只能通过传统的本地方式来解决此问题了。这些方法包括：

1、禁止RealServer响应对VIP的ARP请求；

2、在RealServer上隐藏VIP，以使得它们无法获知网络上的ARP请求；

3、基于“透明代理（Transparent Proxy）”或者“fwmark （firewall mark）”；

4、禁止ARP请求发往RealServers；

传统认为，解决ARP问题可以基于网络接口，也可以基于主机来实现。Linux采用了基于主机的方式，因为其可以在大多场景中工作良好，但LVS却并不属于这些场景之一，因此，过去实现此功能相当麻烦。现在可以通过设置arp\_ignore和arp\_announce，这变得相对简单的多了。

Linux 2.2和2.4（2.4.26之前的版本）的内核解决“ARP问题”的方法各不相同，且比较麻烦。幸运的是，2.4.26和2.6的内核中引入了两个新的调整ARP栈的标志（device flags）：arp\_announce和arp\_ignore。基于此，在DR/TUN的环境中，所有IPVS相关的设定均可使用arp\_announce=2和arp\_ignore=1/2/3来解决“ARP问题”了。

arp\_annouce：Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface；

0 - (default) Use any local address, configured on any interface.

1 - Try to avoid local addresses that are not in the target's subnet for this interface.

2 - Always use the best local address for this target.

arp\_ignore: Define different modes for sending replies in response to received ARP requests that resolve local target IP address.

0 - (default): reply for any local target IP address, configured on any interface.

1 - reply only if the target IP address is local address configured on the incoming interface.

2 - reply only if the target IP address is local address configured on the incoming interface and both with the sender's IP address are part from same subnet on this interface.

3 - do not reply for local address configured with scope host, only resolutions for golbal and link addresses are replied.

4-7 - reserved

8 - do not reply for all local addresses

在RealServers上，VIP配置在本地回环接口lo上。如果回应给Client的数据包路由到了eth0接口上，则arp通告或请应该通过eth0实现，因此，需要在sysctl.conf文件中定义如下配置：

#vim /etc/sysctl.conf

net.ipv4.conf.eth0.arp\_ignore = 1

net.ipv4.conf.eth0.arp\_announce = 2

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.all.arp\_announce = 2

以上选项需要在启用VIP之前进行，否则，则需要在Drector上清空arp表才能正常使用LVS。

到达Director的数据包首先会经过PREROUTING，而后经过路由发现其目标地址为本地某接口的地址，因此，接着就会将数据包发往INPUT(LOCAL\_IN HOOK)。此时，正在运行内核中的ipvs（始终监控着LOCAL\_IN HOOK）进程会发现此数据包请求的是一个集群服务，因为其目标地址是VIP。于是，此数据包的本来到达本机(Director)目标行程被改变为经由POSTROUTING HOOK发往RealServer。这种改变数据包正常行程的过程是根据IPVS表(由管理员通过ipvsadm定义)来实现的。

如果有多台Realserver，在某些应用场景中，Director还需要基于“连接追踪”实现将由同一个客户机的请求始终发往其第一次被分配至的Realserver，以保证其请求的完整性等。其连接追踪的功能由Hash table实现。Hash table的大小等属性可通过下面的命令查看：

# ipvsadm -lcn

为了保证其时效性，Hash table中“连接追踪”信息被定义了“生存时间”。LVS为记录“连接超时”定义了三个计时器：

1、空闲TCP会话；

2、客户端正常断开连接后的TCP会话；

3、无连接的UDP数据包（记录其两次发送数据包的时间间隔）；

上面三个计时器的默认值可以由类似下面的命令修改，其后面的值依次对应于上述的三个计时器：

# ipvsadm --set 28800 30 600

数据包在由Direcotr发往Realserver时，只有目标MAC地址发生了改变(变成了Realserver的MAC地址)。Realserver在接收到数据包后会根据本地路由表将数据包路由至本地回环设备，接着，监听于本地回环设备VIP上的服务则对进来的数据库进行相应的处理，而后将处理结果回应至RIP，但数据包的原地址依然是VIP。

ipvs的持久连接：

无论基于什么样的算法，只要期望源于同一个客户端的请求都由同一台Realserver响应时，就需要用到持久连接。比如，某一用户连续打开了三个telnet连接请求时，根据RR算法，其请求很可能会被分配至不同的Realserver，这通常不符合使用要求。

Director脚本:

#!/bin/bash

#

# LVS script for VS/DR

#

. /etc/rc.d/init.d/functions

#

VIP=192.168.0.210

RIP1=192.168.0.221

RIP2=192.168.0.222

PORT=80

#

case "$1" in

start)

/sbin/ifconfig eth0:1 $VIP broadcast $VIP netmask 255.255.255.255 up

/sbin/route add -host $VIP dev eth0:1

# Since this is the Director we must be able to forward packets

echo 1 > /proc/sys/net/ipv4/ip\_forward

# Clear all iptables rules.

/sbin/iptables -F

# Reset iptables counters.

/sbin/iptables -Z

# Clear all ipvsadm rules/services.

/sbin/ipvsadm -C

# Add an IP virtual service for VIP 192.168.0.219 port 80

# In this recipe, we will use the round-robin scheduling method.

# In production, however, you should use a weighted, dynamic scheduling method.

/sbin/ipvsadm -A -t $VIP:80 -s wlc

# Now direct packets for this VIP to

# the real server IP (RIP) inside the cluster

/sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -g -w 1

/sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -g -w 2

/bin/touch /var/lock/subsys/ipvsadm &> /dev/null

;;

stop)

# Stop forwarding packets

echo 0 > /proc/sys/net/ipv4/ip\_forward

# Reset ipvsadm

/sbin/ipvsadm -C

# Bring down the VIP interface

/sbin/ifconfig eth0:1 down

/sbin/route del $VIP

/bin/rm -f /var/lock/subsys/ipvsadm

echo "ipvs is stopped..."

;;

status)

if [ ! -e /var/lock/subsys/ipvsadm ]; then

echo "ipvsadm is stopped ..."

else

echo "ipvs is running ..."

ipvsadm -L -n

fi

;;

\*)

echo "Usage: $0 {start|stop|status}"

;;

esac

RealServer脚本:

#!/bin/bash

#

# Script to start LVS DR real server.

# description: LVS DR real server

#

. /etc/rc.d/init.d/functions

VIP=192.168.0.219

host=`/bin/hostname`

case "$1" in

start)

# Start LVS-DR real server on this machine.

/sbin/ifconfig lo down

/sbin/ifconfig lo up

echo 1 > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce

/sbin/ifconfig lo:0 $VIP broadcast $VIP netmask 255.255.255.255 up

/sbin/route add -host $VIP dev lo:0

;;

stop)

# Stop LVS-DR real server loopback device(s).

/sbin/ifconfig lo:0 down

echo 0 > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo 0 > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo 0 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 0 > /proc/sys/net/ipv4/conf/all/arp\_announce

;;

status)

# Status of LVS-DR real server.

islothere=`/sbin/ifconfig lo:0 | grep $VIP`

isrothere=`netstat -rn | grep "lo:0" | grep $VIP`

if [ ! "$islothere" -o ! "isrothere" ];then

# Either the route or the lo:0 device

# not found.

echo "LVS-DR real server Stopped."

else

echo "LVS-DR real server Running."

fi

;;

\*)

# Invalid entry.

echo "$0: Usage: $0 {start|status|stop}"

exit 1

;;

esac