链接：http://www.178linux.com/64663

Q1：请描述网桥、集线器、二层交换机、三层交换机、路由器的功能、使用场景与区别。

网桥：网桥（Bridge）是早期的两端口二层网络设备，用来连接不同网段。网桥的两个端口分别有一条独立的交换信道，不是共享一条背板总线，可隔离冲突域。网桥比集线器（Hub）性能更好，集线器上各端口都是共享同一条背板总线的。后来，网桥被具有更多端口、同时也可隔离冲突域的交换机（Switch）所取代。

集线器：集线器的英文称为“Hub”。“Hub”是“中心”的意思，集线器的主要功能是对接收到的信号进行再生整形放大，以扩大网络的传输距离，同时把所有节点集中在以它为中心的节点上。它工作于OSI(开放系统互联参考模型)参考模型第一层，即“物理层”。集线器与网卡、网线等传输介质一样，属于局域网中的基础设备，采用CSMA/CD（即带冲突检测的载波监听多路访问技术)介质访问控制机制。集线器每个接口简单的收发比特，收到1就转发1，收到0就转发0，不进行碰撞检测。

二层交换机：二层交换机工作于OSI模型的第2层（数据链路层），故而称为二层交换机。二层交换技术的发展已经比较成熟，二层交换机属数据链路层设备，可以识别数据包中的MAC地址信息，根据MAC地址进行转发，并将这些MAC地址与对应的端口记录在自己内部的一个地址表中。

三层交换机：三层交换机就是具有部分路由器功能的交换机，三层交换机的最重要目的是加快大型局域网内部的数据交换，所具有的路由功能也是为这目的服务的，能够做到一次路由，多次转发。对于数据包转发等规律性的过程由硬件高速实现，而像路由信息更新、路由表维护、路由计算、路由确定等功能，由软件实现。三层交换技术就是二层交换技术+三层转发技术。传统交换技术是在OSI网络标准模型第二层——数据链路层进行操作的，而三层交换技术是在网络模型中的第三层实现了数据包的高速转发，既可实现网络路由功能，又可根据不同网络状况做到最优网络性能。

路由器：路由器（Router），是连接因特网中各局域网、广域网的设备，它会根据信道的情况自动选择和设定路由，以最佳路径，按前后顺序发送信号。 路由器是互联网络的枢纽，"交通警察"。目前路由器已经广泛应用于各行各业，各种不同档次的产品已成为实现各种骨干网内部连接、骨干网间互联和骨干网与互联网互联互通业务的主力军。路由和交换机之间的主要区别就是交换机发生在OSI参考模型第二层（数据链路层），而路由发生在第三层，即网络层。这一区别决定了路由和交换机在移动信息的过程中需使用不同的控制信息，所以说两者实现各自功能的方式是不同的。

Q2：IP地址的分类有哪些？子网掩码的表示形式及其作用；

IP地址分类

A类IP地址：地址范围0.0.0.0 -- 127.0.0.0，默认子网掩码255.0.0.0

B类IP地址：地址范围128.0.0.0 -- 191.255.0.0，默认子网掩码255.255.0.0

C类IP地址：地址范围192.0.0.0 -- 223.255.255.0，默认子网掩码255.255.255.0

D类是组播地址：地址范围224.0.0.0 -- 239.255.255.255

E类作为保留研究：地址范围240.0.0.0 -- 255.255.255.255

保留私网地址：

A类：10.0.0.0 -- 10.255.255.255

B类：172.16.0.0 -- 172.31.255.255

C类：192.168.0.0 -- 192.168.255.255

特殊地址：

本地还回地址：127.0.0.1 -- 127.0.0.255

主机请求无法被分配时候，自己生成的地址：169.254.0.0

子网掩码的表示形式及其作用

子网掩码用二进制方式表示地话，也是一个32位的数字。它对应IP地址网络标识部分的位全部为“1”，对应IP地址主机标识的部分则全部为“0”。由此，一个IP地址可以不再受限于自己的类别，而是可以用这样的子网掩码自由地定位自己的网络标识长度。当然，子网掩码必须是IP地址的首位连续的“1”。对于子网掩码，目前有两种表示方式：一种是用32位数字表示；另一种，则是在每个IP地址后面追加网络地址的位数，用“/”隔开。

子网掩码作用：用来区分IP地址的网络号与主机号；当TCP/IP网络上的主机相互通信时，就可以利用子网掩码得知这些主机是否在相同的网络区段内；用来将网络分割为多个子网。

Q3：计算机网络的分成模型有哪些（OSI模型和TCP/IP模型），每一层的功能及涉及到的物理设备有哪些。



Q4：如何将Linux主机接入到TCP/IP网络，请描述详细的步骤。（手动指定的方式）

* 输入命令ifconfig -a查看设备名称；
* 输入命令ifconfig DEVICENAME IP进行IP地址设定；
* 输入命令vim /etc/sysconfig/network-script/DEVICENAME修改脚本；
* BOOTPROTO=static
* ONBOOT=yes
* IPADDR
* NETMASK
* GATEWAY
* DNS1
* 输入命令systemctl restart network.service进行服务重载；

Q5：为Linux主机配置网络信息的方式有哪些，请描述各个过程。

临时有效：使用命令ip或者ifconfig进行修改，立即生效；

永久生效：修改网卡配置脚本文件，需要重启服务或者ifdown/ifup对网卡单独操作；

使用图形界面setup--网络配置，需要重启服务生效；

Q6：写一个脚本，使用ping命令探测172.16.250.1-172.16.250.254之间的所有主机的在线状态；在线的主机使用绿色显示；不在线的主使用红色显示；

#!/bin/bash

#

trap 'exit 1' INT

declare -i i=1

for i in(seq 254);do

    if ping 172.16.250.$i -c 1 -W 1 &> /dev/null;then

        echo -e "\033[0;32;1m 172.16.250.$i is up! \033[0m"

    else

        echo -e "\033[0;31;1m 172.16.250.$i is down! \033[0m"

    fi

    let i++

done

Q7：详细描述每个网络接口的配置文件中各个参数的含义和其所对应的值；

ifcfg-IFACE配置文件参数：

    DEVICE：此配置文件对应的设备的名称；

    ONBOOT：在系统引导过程中，是否激活此接口；

    UUID：此设备的惟一标识；

    IPV6INIT：是否初始化IPv6；

    BOOTPROTO：激活此接口时使用什么协议来配置接口属性，常用的有dhcp、bootp、static、none；

    TYPE：接口类型，常见的有Ethernet, Bridge；

    DNS1：第一DNS服务器指向；

    DNS2：备用DNS服务器指向；

    DOMAIN：DNS搜索域；

    IPADDR： IP地址；

    NETMASK：子网掩码；CentOS 7支持使用PREFIX以长度方式指明子网掩码；

    GATEWAY：默认网关；

    USERCTL：是否允许普通用户控制此设备；

    PEERDNS：如果BOOTPROTO的值为“dhcp”，是否允许dhcp server分配的dns服务器指向覆盖本地手动指定的DNS服务器指向；默认为允许；

    HWADDR：设备的MAC地址；    #不能随便修改，出厂的时候厂商给的，在文件中删除就好了

    NM\_CONTROLLED：是否使用NetworkManager服务来控制接口；    #在CentOS 6及以前不启用，一般用network）

Q8：如何给网络接口配置多个地址，有哪些方式？

#临时生效：

使用ip / ifconfig输入命令：

~]# ip addr add IP/mask dev eth0:0

~]# ifconfig IFACE\_LABEL IPADDR/NETMASK            #其中IFACE\_LABEL：eth0:0, eth0:1；

#修改配置文件：

DEVICE=IFACE\_LABEL

BOOTPROTO:网上别名不支持动态获取地址, static, none；

Q9：常用的网络管理类工具有哪些，并用示例形式描述他们的使用方法。

ifconfig命令：接口及地址查看和管理

    ifconfig  [INTERFACE]

    # ifconfig -a：显示所有接口，包括inactive状态的接口；

    ifconfig interface [aftype] options | address ...

    # ifconfig  IFACE  IP/MASK  [up|down]

    # ifconfig  IFACE  IP  netmask  NETMASK

    options：

    [-]promisc    混杂模式；

    ~]# ifconfig eth1 promisc    打开混杂模式；

    ~]# ifconfig eth1 -promisc    关闭混杂模式；

route命令：路由查看及管理

    查看：

    # route  -n

    添加：

    route  add  [-net|-host]  target  [netmask  Nm]  [gw GW]  [[dev] If]   网络路由/主机路由

    示例：route add -net  10.0.0.0/8  gw  192.168.10.1  dev  eth1

          route add  -net  0.0.0.0/0.0.0.0  gw 192.168.10.1

          route add  default  gw 192.168.10.1      表示为默认网关

          【配置单个主机】options改为 -host即可

    删除：

    route  del  [-net|-host] target  [gw Gw]  [netmask Nm]  [[dev] If]

    示例：route  del  -net  10.0.0.0/8  gw 192.168.10.1

          route  del  default

netstat命令：

    显示路由表：netstat  -rn

    -r：显示内核路由表

    -n：数字格式

    显示网络连接：

    netstat  [--tcp|-t]  [--udp|-u]  [--udplite|-U]  [--sctp|-S]  [--raw|-w]  [--listening|-l]  [--all|-a]  [--numeric|-n]   [--extend|-e[--extend|-e]]  [--program|-p]

    -t：TCP协议的相关连接，连接均有其状态；FSM（Finate State Machine）；

    -u：UDP相关的连接

    -w：raw socket相关的连接

    -l：处于监听状态的连接

    -a：所有状态

    -n：以数字格式显示IP和Port；

    -e：扩展格式

    -p：显示相关的进程及PID；

    常用组合：

    -tan,  -uan,  -tnl,  -unl,  -tunlp

显示接口的统计数据：

    netstat    {--interfaces|-I|-i}    [iface]   [--all|-a]   [--extend|-e]   [--verbose|-v]   [--program|-p]  [--numeric|-n]

    所有接口：

    netstat  -i

    指定接口：

    netstat  -I<IFace>

    ifup/ifdown命令：

    注意：通过配置文件/etc/sysconfig/network-scripts/ifcfg-IFACE来识别接口并完成配置；

ip命令：

    ip [ OPTIONS ] OBJECT { COMMAND | help }

    OBJECT := { link | addr | route | netns  }

    注意： OBJECT可简写，各OBJECT的子命令也可简写；

    ip  OBJECT：

    ip link： network device configuration        网络设备配置；

    ip  link  set - change device attributes        与ifconfig很相似，用于设置某属性属于什么状态；

    <[dev] NAME (default)>：指明要管理的设备，dev关键字可省略；

    up和down：启用或禁用接口；  ~]# ip link set [dev] eth1 up

    multicast on或multicast off：启用或禁用多播功能；

    name NAME：重命名网络接口；

    mtu NUMBER：设置MTU的大小，以太网接口默认为1500；

    netns PID：ns为namespace，用于将接口移动到指定的网络名称空间；

    ip  link  show  - display device attributes

    ip  link  help -  显示简要使用帮助；

    ip netns：  - manage network namespaces.

    ip  netns  list：列出所有的netns

    ip  netns  add  NAME：创建指定的netns

    ip  netns  del  NAME：删除指定的netns

    ip  netns   exec  NAME  COMMAND：在指定的netns中运行命令

ip address - protocol address management.    用于协议地址管理，简言之用于管理IP地址；

    ip address add - add new protocol address        添加新IP地址；

    ip  addr  add  IFADDR  dev  IFACE

    [label NAME]：为额外添加的地址（主地址以外）指明接口别名；

    [broadcast ADDRESS]：广播地址；会根据IP和NETMASK自动计算得到；

    [scope SCOPE\_VALUE]：（了解，不记也没关系）scope表示作用域；

    global：全局可用；（大多数情况下，就使用这个）

    link：接口可用；

    host：仅本机可用；

    ip address delete - delete protocol address        删除IP地址；

    ip addr  delete  IFADDR  dev  IFACE

    ip address show - look at protocol addresses    查看IP地址；

    ip  addr   list  [IFACE]：显示接口的地址；

    ip address flush - flush protocol addresses        删除IP地址，清空所有的IP地址；

    ip  addr  flush  dev  IFACE

ip route - routing table management            #用于路由表管理命令；

    ip route add - add new route                #添加路由；

    ip route change - change route                #修改路由；

    ip route replace - change or add new one    #（有老的）修改路由，（没有老的）添加路由；

    ip  route   add  TYPE PREFIX  via GW  [dev  IFACE]  [src SOURCE\_IP]        #PREFIX指的是长度的掩码；

    示例：

    # ip route add 192.168.0.0/24  via 10.0.0.1  dev eth1 src  10.0.20.100

    # ip  route  add default  via  GW

    ip route delete - delete route                #删除路由；

    ip  route  del  TYPE PRIFIX

    示例：

    # ip  route delete  192.168.1.0/24

    ip route show - list routes                    #显示路由信息；

    TYPE PRIFIX

    ip route flush - flush routing tables        #清空路由表；

    TYPE  PRIFIX

    ip route get - get a single route            #获取（指定的）单条路由；（了解即可）

    ip  route  get  TYPE PRIFIX

    示例：ip route  get  192.168.0.0/24

ss命令：

    ss  [options]  [ FILTER ]

    选项：

    -t：TCP协议的相关连接

    -u：UDP相关的连接

    -w：raw socket相关的连接

    -l：监听状态的连接

    -a：所有状态的连接

    -n：数字格式

    -p：相关的程序及其PID

    -e：扩展格式信息

    -m：内存用量

    -o：计时器信息

    FILTER := [ state TCP-STATE ]  [ EXPRESSION ]

    TCP的常见状态：

    TCP FSM：

    LISTEN：监听

    ESTABLISEHD：建立的连接

    FIN\_WAIT\_1：

    FIN\_WAIT\_2：

    SYN\_SENT：

    SYN\_RECV：

    CLOSED：

    EXPRESSION：

    dport =

    sport =

    示例：'( dport = :22 or sport = :22)'

    ~]# ss   -tan    '(  dport = :22 or sport = :22  )'

    ~]# ss  -tan  state  ESTABLISHED

Q10：Linux系统软件包管理方法（安装、升级、卸载等操作）有哪些，以及如何管理的。

rpm命令：rpm  [OPTIONS]  [PACKAGE\_FILE]

安装：

    rpm {-i|--install} [install-options] PACKAGE\_FILE ...

    rpm  -ivh  PACKAGE\_FILE ...

    GENERAL OPTIONS：

    -v：verbose，详细信息

    -vv：更详细的输出

    [install-options]：

    -h：hash marks输出进度条；每个#表示2%的进度；

    --test：测试安装，检查并报告依赖关系及冲突消息等；

    --nodeps：忽略依赖关系；不建议；

    --replacepkgs：重新安装

    注意：rpm可以自带脚本；

    四类：--noscripts

    preinstall：安装过程开始之前运行的脚本，%pre ， --nopre

    postinstall：安装过程完成之后运行的脚本，%post , --nopost

    preuninstall：卸载过程真正开始执行之前运行的脚本，%preun, --nopreun

    postuninstall：卸载过程完成之后运行的脚本，%postun , --nopostun

    --nosignature：不检查包签名信息，不检查来源合法性；

    --nodigest：不检查包完整性信息；

升级：

    rpm {-U|--upgrade} [install-options] PACKAGE\_FILE ...

    rpm {-F|--freshen} [install-options] PACKAGE\_FILE ...

    -U：升级或安装；

    -F：升级

    rpm  -Uvh PACKAGE\_FILE ...

    rpm  -Fvh PACKAGE\_FILE ...    纯粹的升级操作；

    --oldpackage：降级；

    --force：强制升级；

    注意：(1) 不要对内核做升级操作；Linux支持多内核版本并存，因此，直接安装新版本内核；

    (2) 如果某原程序包的配置文件安装后曾被修改过，升级时，新版本的程序提供的同一个配置文件不会覆盖原有版本的配置文件，而是把新版本的配置文件重命名(FILENAME.rpmnew)后提供；

卸载：

    rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--test] PACKAGE\_NAME ...

    卸载时是PACKAGE\_NAME，包名即可；

    --allmatches：卸载所有匹配指定名称的程序包的各版本；

    --nodeps：忽略依赖关系

    --test：测试卸载，dry run模式

查询：

    rpm {-q|--query} [select-options] [query-options]

    [select-options]

    PACKAGE\_NAME：查询指定的程序包是否已经安装，及其版本；

    -a, --all：查询所有已经安装过的包；

    -f  FILE：查询指定的文件FILE由哪个程序包安装生成；

    -p, --package PACKAGE\_FILE：用于实现对未安装的程序包执行查询操作；

    --whatprovides CAPABILITY：查询指定的CAPABILITY由哪个程序包提供；

    --whatrequires CAPABILITY：查询指定的CAPABILITY被哪个包所依赖；

    e.g.

    ~]# rpm -q --whatprovides bash        查询bash安装时的包的名称；

    bash-4.2.46-19.el7.x86\_64

    ~]# rpm -q --whatrequires bash        查询bash被哪些程序包所依赖；

    dracut-033-359.el7.x86\_64

    initscripts-9.49.30-1.el7.x86\_64

    rsyslog-7.4.7-12.el7.x86\_64

    lvm2-2.02.130-5.el7.x86\_64

    [query-options]

    --changelog：查询rpm包的changlog；

    -l, --list：程序安装生成的所有文件列表；

    -i, --info：程序包相关的信息，版本号、大小、所属的包组，等；

    -c, --configfiles：查询指定的程序包提供的配置文件；

    -d, --docfiles：查询指定的程序包提供的文档；

    --provides：列出指定的程序包提供的所有的CAPABILITY；

    -R, --requires：查询指定的程序包的依赖关系；

    --scripts：查看程序包自带的脚本片断；（四个脚本：安装前、安装、卸载、卸载后，其中安装前和卸载后可以没有）

用法：

    -qi  PACKAGE, -qf FILE, -qc PACKAGE, -ql PACKAGE, -qd PACKAGE

    -qpi  PACKAGE\_FILE, -qpl PACKAGE\_FILE, -qpc PACKAGE\_FILE, ...

校验：

    rpm {-V|--verify} [select-options] [verify-options]

    防止安装包在发布的时候，被做了修改，开启后门程序之类的；

    S file Size differs        文件大小发生改变；

    M Mode differs (includes permissions and file type)  权限发生改变；

    5 digest (formerly MD5 sum) differs  MD5码发生改变，文件的内容完整性发生改变；

    D Device major/minor number mismatch  主次设备号不匹配；

    L readLink(2) path mismatch              readLink的路径不匹配；

    U User ownership differs              属主修改了；

    G Group ownership differs              属组修改了；

    T mTime differs                          最近一次的时间戳修改了；

    P caPabilities differ                  caPabilities发生改变；

yum命令的用法：yum [options] [command] [package ...]

显示仓库列表：

repolist [all|enabled|disabled]

显示程序包：

    list

    # yum list [all | glob\_exp1] [glob\_exp2] [...]

    # yum list {available|installed|updates} [glob\_exp1] [...]

安装程序包：

    install package1 [package2] [...]

    reinstall package1 [package2] [...]  (重新安装)

升级程序包：

    update [package1] [package2] [...]

    downgrade package1 [package2] [...] (降级)

检查可用升级：

    check-update

卸载程序包：        依赖于次程序包的也都会被卸载；

    remove | erase package1 [package2] [...]

查看程序包information：        显示出类似于rpm -qi的结果；

    info [...]

查看指定的特性(可以是某文件)是由哪个程序包所提供：            显示出类似于rpm -qf的结果；

    provides | whatprovides feature1 [feature2] [...]

清理本地缓存：    因为yum下载之后，程序包会自动删除，元数据不会被删除，用这个命令手动删除；

    clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]

构建缓存：从各仓库下载创建，无需自己去做，每次使用yum命令的时候会自动做的；

    makecache

搜索：

    search string1 [string2] [...]

    以指定的关键字搜索程序包名及summary信息；

查看指定包所依赖的capabilities：

    deplist package1 [package2] [...]

查看yum事务历史：

    history [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|new|sync|stats]

安装及升级本地程序包：

    yum localinstall rpmfile1 [rpmfile2] [...]

           (maintained for legacy reasons only - use install)

yum localupdate rpmfile1 [rpmfile2] [...]

           (maintained for legacy reasons only - use update)

包组管理的相关命令：

    yum groupinstall group1 [group2] [...]        安装指定包组；组名可能要加“”；

    yum groupupdate group1 [group2] [...]

    yum grouplist [hidden] [groupwildcard] [...]    显示指定包组；

    yum groupremove group1 [group2] [...]

    yum groupinfo group1 [...]

Q11：如何使用发行版光盘作为yum repository，请描述该过程。

(1)挂载光盘至某目录

~]# mount -r /dev/cdrom /media/cdrom

(2)创建配置文件

~]# vim /etc/yum.repos.d/local.repo

[CentOS7]

name=CentOS7

baseurl=file:///media/cdrom

enable=1

gpgcheck=0

Q12：写一个脚本，完成以下功能

* (1) 假设某目录(/etc/rc.d/rc3.d/)下分别有K开头的文件和S开头的文件若干；
* (2) 显示所有以K开头的文件的文件名，并且给其附加一个stop字符串；
* (3) 显示所有以S开头的文件的文件名，并且给其附加一个start字符串；
* (4) 分别统计S开头和K开头的文件各有多少；

#!/bin/bash

#

declare -i s=0

declare -i k=0

for i in $(ls /etc/rc.d/rc3.d/);do

    if [[ $i == S\* ]];then

        echo $i.start

        s+=1

    fi

    if [[ $i == K\* ]];then

                echo $i.stop

                k+=1

    fi

done

echo -e "S start:$s\nK start:$k"

Q13：写一个脚本，完成以下功能

* (1) 脚本能接受用户名作为参数；
* (2) 计算此些用户的ID之和；

#!/bin/bash

#

[ $# -eq 0 ] && echo -e "User name(s) needed!\nPlease try again!" && exit 1

declare -i sum=0

for i in $@;do

        if id $i &> /dev/null;then

                sum+=$(id -u $i)

        else

                echo "$i not exits!"

        fi

done

echo "Exiting users' ID sum:$sum"

Q14：写一个脚本

* (1) 传递一些目录给此脚本；
* (2) 逐个显示每个目录的所有一级文件或子目录的内容类型；
* (3) 统计一共有多少个目录；且一共显示了多少个文件的内容类型；

#!/bin/bash

#

if [ $# -lt 1 ];then

        echo -e "At least one arg!\nPlease try again!"

        exit 1

fi

declare -i dir=0

declare -i file=0

for i in $@;do

        if ls $i &> /dev/null;then

                for j in $(ls $i);do

                        if [ -d $i/$j ];then

                                let dir++

                        else [ -f $i/$j ]

                                let file++

                        fi

                done

        else

                echo "$i not exits!"

        fi

done

echo "directory number:$dir,file number:$file"

Q15：写一个脚本：通过命令行传递一个参数给脚本，参数为用户名，如果用户的id号大于等于500，则显示此用户为普通用户；

#!/bin/bash

#

if [ $# -lt 1 ];then

        echo "One arg is needed!"

        exit 1

fi

if id $1 &> /dev/null;then

        if [ `id -u $1` -ge 500 ];then

                echo "$1 is a nomal user."

        elif [ `id -u $1` -ge 1 ];then

                echo "$1 is a system user."

        else

                echo "$1 is a root user."

        fi

else

        echo "$1 not exits."

fi

Q16：写一个脚本

* (1) 添加10用户user1-user10；密码同用户名；
* (2) 用户不存在时才添加；存在时则跳过；
* (3) 最后显示本次共添加了多少用户；

#!/bin/bash

#

declare -i sum=0

for i in {1..10};do

        id user$i &> /dev/null && continue || useradd user$i && echo "user$i" | passwd --stdin $user$i

        sum+=1

done

echo "$sum users were added this time."

Q17：写一脚本，用ping命令测试172.16.250.20-172.16.250.100以内有哪些主机在线，将在线的显示出来；

#!/bin/bash

#

trap 'exit 1' INT

for ((i=20;i<=100;i++));do

    if ping 172.16.250.$i -c 1 -W 1 &> /dev/null;then

        echo -e "172.16.250.$i is up!"

    fi

done

Q18：打印九九乘法表；

#!/bin/bash

#

for j in {1..9}; do

        for i in $(seq 1 $j); do

                echo -n -e "${i}X${j}=$[${i}\*${j}]\t"

        done

        echo

done