Questions   
1. What are the different ways to check the load average on a system?   
vmstat, top, uptime, w, procinfo   
================================   
================================   
  
Bonus - Describe the 3 values that top/uptime shows   
1-minute, 5-minute and 15-minute load averages   
================================   
================================   
  
2. What are the different running states of a SOLARIS system?   
1,2, and 3   
================================   
================================   
  
3. How do you check CPU and MEMORY resources on a server?   
Memory: dmesg |grep mem, prtdiag |grep Memory, prtconf -v |grep Mem   
CPU: /usr/sbin/psrinfo -v   
================================   
================================   
  
4. How do you obtain system activity for a particular time frame (Say noon to 10PM)?   
a) Use the command 'sar'   
b) sar consists of three commands that are involved in automatic system activity   
data collection: sadc, sa1, and sa2.   
c) To make sure sadc is run at boot time, the /etc/init.d/perf file must contain   
a command line that writes a record to the daily data file.   
d) The command entry has the following format: /usr/bin/su sys -c   
"/usr/lib/sa/sadc /var/adm/sa/sa`date +%d`"   
e) This entry is already in the /etc/init.d/perf file, but it needs to be   
uncommented.   
f) Put a line into the /var/spool/cron/crontabs/sys file, which calls the shell   
script, sa1. This script invokes sadc and writes to the daily data files,   
/var/adm/sa/sa<dd>. The sa1 script gets installed with the other sar packages   
and has the following syntax: /usr/lib/sa/sa1 [t n]   
g) The syntax for the sar command is as follows: sar [-aAbcdgkmpqruvwy] [-o   
<outputfile>] [t n ]   
h) So in answer to the original question the command to obtain system activity   
from 12:00 PM to 10:00 PM is as follows: sar -s 12 -e 22 -i 3600 -A   
================================   
================================   
  
5. What does an init 5 do?   
shutdown the system, it will sync the file system first.   
================================   
================================   
  
6. How do you reset the root password on a server? No one has the password or   
has forgotten it. SUDO is not configured on the server as well.   
a) Insert Solaris CD in cd drive and from ok prompt run command: boot cdrom -s   
b) This will take you single user mode   
# fsck /dev/rdsk/c0t0d0s0.   
Answer y to clear all.   
c) mount /dev/dsk/c0t0d0s0 /a   
d) cd /a/etc   
e) TERM=sun   
f) export TERM   
g) vi /a/etc/shadow   
Remove password (between the first two colons i.e..,   
root:WYlPW5T2EyiU6:13750::::::) from password field of root and save file with   
wq!   
h) cd /   
i) umount /a   
j) init 6   
You will be prompted for password for root.   
================================   
================================   
  
7. How do you check disk usage. How do you trouble shoot a high disk usage issue   
(Available disk space is at 2% and could crash the application)   
First see which partiton is full   
du -hk   
To find out which files/folders are taking up the most space   
/du -dk / | sort -n   
To delete files older than x number of days in the current working directory and   
below, the safe way is:  
   
For log files older than 5 days  
  
find /opt/app/logs/ -name \*.log -mtime +5 -exec ls -tl {} \;   
  
find /opt/app/logs/ -name \*.log -mtime +5 -exec rm -f {} \;  
  
  
  
For log files newer than 5 days  
  
find /opt/app/logs/ -name \*.log -mtime -5 -exec ls -tl {} \;  
  
find /opt/app/logs/ -name \*.log -mtime -5 -exec rm -f {} \;  
================================   
================================   
  
8. How do you check the ports in use on a server?   
netstat -an   
================================   
================================   
  
9. What is NDD?   
Make the changes to the running system.   
# ndd -set /dev/hme adv\_100hdx\_cap 0   
# ndd -set /dev/hme adv\_100fdx\_cap 1   
# ndd -set /dev/hme adv\_autoneg\_cap 0   
# ndd -get /dev/hme link\_mode   
Interpretation:   
0 -- half-duplex   
1 -- full-duplex   
# ndd -get /dev/hme link\_speed   
Interpretation:   
0 -- 10 Mbit   
1 -- 100 Mbit   
1000 -- 1 Gbit   
================================   
================================   
  
10. What is garbage collection in Java?   
When an object is no longer referenced by the program, the heap space it   
occupies must be recycled so that the space is available for subsequent new   
objects. The garbage collector must somehow determine which objects are no   
longer referenced by the program and make available the heap space occupied by   
such unreferenced objects. In the process of freeing unreferenced objects, the   
garbage collector must run any finalizers of objects being freed.   
================================   
================================   
  
  
======================================   
NETWORKING /////////////////////////   
======================================   
  
======================================   
Solaris Networking Commands:   
======================================   
  
Route Configuration   
===================   
route add net 128.50.0.0 128.50.1.6 1   
/\* Add a route to the routing table \*/   
=======================================   
route change 128.50.0.0 128.50.1.5   
/\* Changes the destination address for a route \*/   
=======================================   
route delete net 128.50.0.0 128.50.1.6   
/\* Delete a route from the routing table \*/   
=======================================   
route flush   
/\* Flush the routing table, which will remove all entries \*/   
=======================================   
route get [hostname]   
/\* Which interface will be used to contact hostname \*/   
=======================================   
route monitor   
/\* Monitor routing table lookup misses and changes \*/   
=======================================   
Network Information   
arp -a   
/\* Ethernet address arp table \*/   
=======================================   
arp -d myhost   
/\* Delete an ethernet address arp table entry \*/   
=======================================   
lsof [-iTCP@10.20](mailto:-iTCP@10.20)   
/\* Display open files for internet address \*/   
=======================================   
named-xfer -z qantas.com.au -f /tmp/allip   
/\* Get All IP Addresses On A DNS Server \*/   
=======================================   
ndd /dev/arp arp\_cache\_report   
/\* Prints ARP table in cache with IP and MAC address \*/   
=======================================   
netstat -a | grep EST | wc -l   
/\* Displays number active established connections to the localhost \*/   
=======================================   
netstat -a | more   
/\* Show the state of all the sockets on a machine \*/   
=======================================   
netstat -i   
/\* Show the state of the interfaces used for TCP/IP traffice \*/   
=======================================   
netstat -k hme0   
/\* Undocumented netstat command \*/   
=======================================   
netstat -np   
/\* Similar to arp -a without name resolution \*/   
=======================================   
netstat -r   
/\* Show the state of the network routing table for TCP/IP traffic \*/   
=======================================   
netstat -rn   
/\* Displays routing information but bypasses hostname lookup. \*/   
=======================================   
snoop -S -ta [machine]   
/\* Snoop for network packets and get size and time stamp entries. \*/   
=======================================   
traceroute <ipaddress>   
/\* Follow the route to the ipaddress \*/   
=======================================

======================================   
Linux Networking Commands:   
======================================   
  
======================================   
Basic Linux Network Commands:   
======================================   
This category contains the most basic network commands available on Linux platform.   
======================================   
w   
Shows who is currently logged in and where they are logged in from.   
======================================   
who   
This also shows who is on the server in an shell.   
======================================   
netstat   
Shows all current network connections.   
======================================   
netstat -an   
Shows all connections to the server, the source and destination ips and ports.   
======================================   
netstat -rn   
Shows routing table for all ips bound to the server.   
======================================   
netstat -an |grep :80 |wc -l   
Show how many active connections there are to apache (httpd runs on port 80)   
======================================   
top   
Shows live system processes in a formatted table, memory information, uptime and   
other useful info.   
======================================   
While in top, Shift + M to sort by memory usage or Shift + P to sort by CPU usage.   
======================================   
top -u root   
Show processes running by user root only.   
======================================   
route -n   
Shows routing table for all ips bound to the server.   
======================================   
route add default gw my\_computer   
Add a default gateway to my\_computer.   
======================================   
nslookup yahoo.com   
Query your default domain name server (DNS) for an Internet name (or IP number)   
host\_to\_find.   
======================================   
traceroute yahoo.com   
Have a look how you messages travel to yahoo.com   
======================================   
tracepath yahoo.com   
Performs a very similar function to traceroute.   
======================================   
ifconfig   
Display info on the network interfaces.   
======================================   
ifconfig -a   
Display into on all network interfaces on server, active or inactive.   
======================================   
ifconfig eth0 down   
This will take eth0 (assuming the device exists) down, it won't be able to receive or  
send anything until you put the device back "up" again.   
======================================   
ifconfig eth0 up   
You guessed it. This would take eth0 up and available to receive or send packets.   
======================================   
/sbin/ifconfig eth0 192.168.10.12 netmask 255.255.255.0 broadcast 192.168.10.255   
Assign IP 192.168.10.12, netmask and broadcast address to interface eth0.   
======================================   
ifup eth0   
Will bring eth0 up if it is currently down.   
======================================   
ifdown eth0   
Will bring eth0 down if it is currently up.   
======================================   
ifcfg   
Use ifcfg to configure a particular interface. Simply type ifcfg to get help on using  
this script.   
======================================   
ifcfg eth0 del 192.168.0.1   
This command takes eth0 down and removes the assigned IP 192.168.0.1   
======================================   
ifcfg eth0 add 192.168.0.2   
This command brings eth0 up and assigns the new IP 192.168.0.2   
======================================   
ping   
Sends test packets to a specified server to check if it is responding properly   
======================================   
ping yahoo.com   
Sends echo requests to yahoo.com   
======================================   
mii-tool   
Checks what your duplex settings are.   
======================================   
arp   
Command mostly used for checking existing Ethernet connectivity and IP address   
======================================   
hostname   
Tells the user the host name of the computer they are logged into.   
======================================   
findsmb   
Used to list info about machines that respond to SMB name queries. findsmb with no   
argument would find all machines possible. You can also specify a particular subnet  
to localize search.   
======================================   
host yahoo.com   
Performs a simple lookup of an internet address using DNS.   
======================================   
dig yahoo.com   
The "domain information groper" tool. This example looks up information about   
yahoo.com such as IP.   
======================================   
dig -x 66.94.234.13   
Looks up the address and returns the associated domain name. dig takes a huge number   
of options (at the point of being too many), refer to the manual page for more   
information.   
======================================   
whois   
Used to look up the contact information from the "whois" databases. Also reports IP   
address and name server of domain as well as creation and expiration dates.   
======================================   
ftp   
File transfer protocol. Transfers files to another host (insecure)   
======================================   
rdesktop   
Display remote desktop on Linux Machine. You can use to connect to Windows.   
======================================   
  
======================================   
ADVANCED NETWORK TIPS: ///////////////   
======================================   
  
All the remote network administration related tools and techniques available on Linux  
platform.   
======================================   
ssh   
Secure shell, an alternative but secure to telnet/rsh and all the non-secure methods   
of logging in to remote servers. All connections get encrypted.   
======================================   
ssh username@hostname   
Connect to a remote server by specifying your username and hostname you're logging   
into.   
======================================   
scp <from\_server> <to\_server>   
Secure copy. Allows you to copy files from one computer to another computer, use -r   
to copy recursively.   
======================================   
scp -r jose@remote1:/tmp greg@remote2:/tmp   
Do a recursive scp of /tmp on remote1 server logging in as jose to remote2 server   
/tmp logging in as greg.   
======================================   
scp remote:/home/me/junk/\* .   
This will copy files on the remote machine in the directory "/home/me/junk/" to your   
local computer.   
======================================   
sftp   
Secure ftp, another part of the ssh package. This command is similar to ftp but uses   
an encrypted tunnel to connect to an ftp server and is therefore more secure than   
just plain ftp.   
======================================   
rsync   
An open source utility that provides fast incremental file transfer. Can be   
transferred via ssh.   
======================================   
rsync -av -e ssh remote@server:/home/dir /local/dir   
Rsync command used via ssh to login as default user on remote server to fetch   
/home/dir to local server and path /local/dir.   
======================================   
tcpdump   
Print all the network traffic going through the network. Do a 'man tcpdump' to learn   
more.   
======================================   
tcpdump -v   
Display the verbose output.   
======================================   
tcpdump -D   
Display network interfaces available for the capture.   
======================================   
tcpdump -n   
Display numerical addresses rather than symbolic (DNS) addresses.   
======================================   
tcpdump -i eth0   
Capture the traffic of eth0 interface.   
======================================   
tcpdump udp   
Capture the UDP traffic.   
======================================   
tcpdump -w   
capture.log Send the capture output in a file instead of directly on the screen.   
======================================   
tcpdump -r capture.log   
Read a capture file.   
======================================   
tcpdump port http   
Capture the TCP port 80 traffic.   
======================================   
tcpdump -i eth0 host 66.94.234.13   
Listen to all traffic on interface eth0 going to 66.94.234.13. This troubleshooting   
technique can determine why a web connection is not reaching yahoo.com   
(66.94.234.13).   
======================================   
tcpdump host www.yahoo.com   
Display the packets having "www.openmaniak.com" as their source or destination   
address.   
======================================   
tcpdump src 192.168.1.2 and dst 192.168.1.3 and port ftp   
Display the FTP packets coming from 192.168.1.2 to 192.168.1.3.   
======================================   
nmap   
A very advanced network tool used to query machines (local or remote) as to whether   
they are up and what ports are open on these machines. Download it from   
insecure.org and for additional documentation.   
======================================   
nmap host\_name   
This would query host\_name and report what ports it keeps open.   
======================================   
nc   
Netcat is a networking utility which reads and writes data across network   
connections, using the TCP/IP protocol.   
======================================   
wget   
(GNU Web get) used to download files from the World Wide Web. To archive a single   
web-site.   
======================================   
-m or --mirror -->   
To archive a single website.   
======================================   
-nc -->   
no clobber option to stop wget from overwriting a file if you already have it.   
======================================   
-c or --continue -->   
Continue a file that was unfinished by wget or another program.   
======================================   
Wget   
has a large list of options. Please check the manual pages for more details.   
======================================   
wget http://blog.lxpages.com/ultimate\_linux.html   
This would simply get ultimate\_linux.html from blog.lxpages.com website.   
======================================   
curl   
Another remote downloader similar to wget. This remote downloader is designed to work  
without user interaction and supports a variety of protocols, can upload/download   
and has a large number of tricks/work-arounds for various things. It can access   
dictionary servers (dict), ldap servers, ftp, http, gopher, see the manual page for  
full details.   
======================================   
curl -M   
To access the full manual. There are too many options and variations for examples.   
Please refer to manual for in depth examples and techniques.   
======================================   
curl -u   
username:password -T index.html ftp://ftp.mywebsite.com This uploads index.html to   
ftp.mywebsite.com.   
======================================

======================================   
APACHE SHELL COMMANDS: //////////////   
======================================   
Some of the basic and helpful apache commands.   
======================================   
httpd -v   
Outputs the build date and version of the Apache server.   
======================================   
httpd -l   
Lists compiled in Apache modules.   
======================================   
httpd status   
Only works if mod\_status is enabled and shows a page of active connections.   
======================================   
service httpd restart   
Restarted Apache web server.   
======================================   
ab -n 100 -c 5 http://blog.lxpages.com/linux\_network.html   
Apache benchmark. Great tool for load testing your site. -n 100 will send 100 # of   
requests to blog.lxpages.com in order to benchmark.   
======================================   
-c 5   
is # of concurrency.   
======================================   
  
======================================   
NETWORK CONFIGURATION FILES: ///////   
======================================   
All the network related configuration files on a Linux platform.   
======================================   
/etc   
This directory contains most of the basic Linux system-configuration Files.   
======================================   
/etc/sysconfig   
Contains important system configuration files that are created and maintained by   
various services (including iptables, samba, and most networking services).   
======================================   
/etc/sysconfig/network   
Network configuration file used by the system during the boot process.   
======================================   
/etc/sysconfig/network-scripts   
Configuration files that are run during boot process related to setting up of your   
network.   
======================================   
/etc/xinetd.d   
Contains a set of files, each of which defines a network service that the xinetd   
daemon listens for on a particular port.   
======================================   
/etc/syslogd.conf   
The configuration file for the syslogd daemon. syslogd is the daemon that takes care   
of logging (writing to disk) messages coming from other programs to the system.   
======================================   
/etc/resolv.conf   
Host name resolver configuration file. This configures Linux so that it knows which   
DNS server will be resolving domain names into IP addresses.   
======================================   
/etc/hosts   
Locally resolve node names to IP addresses. This informs Linux of local systems on   
the network which are not handled by the DNS server.   
======================================   
/etc/nsswitch.conf   
System Databases and Name Service Switch configuration file. Looks up /etc/hosts   
first, if host not found then it would query DNS server as defined by   
/etc/resolv.conf   
======================================   
/var   
Contains variable data like system logging files, mail and printer spool directories,  
and transient and temporary files.   
======================================   
/var/log   
Log files from the system and various programs/services, especially login   
(/var/log/wtmp, which logs all logins and logouts into the system) and syslog   
(/var/log/messages, where all kernel and system program message are usually   
stored).   
======================================   
/var/log/messages System logs. The first place you should look at if your system is   
in trouble.   
======================================   
/var/log/utmp   
Active user sessions. This is a data file and as such it can not be viewed normally.   
======================================   
/var/log/wtmp   
Log of all users who have logged into and out of the system. The last command can be   
used to access a human readable form of this file.   
======================================   
  
  
  
======================================   
UBUNTU DEBIAN NETWORK CONFIGURATION FILES:  
======================================   
  
======================================   
File: /etc/network/interfaces   
======================================   
Static IP example:   
auto lo   
iface lo inet loopback   
auto eth0   
iface eth0 inet static   
address 208.88.34.106   
netmask 255.255.255.248   
broadcast 208.88.34.111   
network 208.88.34.104   
gateway 208.88.34.110  
   
======================================   
Dynamic IP (DHCP) example:   
======================================   
auto lo   
iface lo inet loopback   
auto eth0   
iface eth0 inet dhcp   
auto eth1   
iface eth1 inet dhcp   
auto eth2   
iface eth2 inet dhcp   
auto ath0   
iface ath0 inet dhcp   
auto wlan0   
iface wlan0 inet dhcp   
  
======================================   
Interfaces:   
======================================   
\* lo: Loopback interface (network within your system without slowing down for the   
real ethernet based network)   
\* eth0: First ethernet interface card   
\* wlan0: First wireless network interface   
Also see "man interfaces"   
  
======================================   
REDHAT FEDORA CORE NETWORK CONFIGURATION FILES:   
======================================   
The Red Hat configuration tools store the configuration information in the file   
/etc/sysconfig/network.   
They will also allow one to configure routing information.   
  
\* File: /etc/sysconfig/network   
Static IP address Configuration: (Configure gateway address)   
NETWORKING=yes   
HOSTNAME=my-hostname - Hostname is defined here and by command hostname   
FORWARD\_IPV4=true - True for NAT firewall gateways and linux routers.   
False for everyone else - desktops and servers.   
GATEWAY="XXX.XXX.XXX.YYY" - Used if your network is connected to another network or   
the internet.   
  
Static IP configuration. Gateway not defined here for DHCP client.   
OR for DHCP client configuration:   
NETWORKING=yes   
HOSTNAME=my-hostname - Hostname is defined here and by command hostname   
(Gateway is assigned by DHCP server.)   
OR for NIS client configuration:   
NETWORKING=yes   
HOSTNAME=my-hostname - Hostname is defined here and by command hostname   
NISDOMAIN=NISProject1 - NIS domain to attach   
  
\* File (Red Hat/Fedora): /etc/sysconfig/network-scripts/ifcfg-eth0   
(S.u.s.e.: /etc/sysconfig/network/ifcfg-eth-id-XX:XX:XX:XX:XX)   
This file used by the command scripts ifup and ifdown   
Static IP address configuration:  
   
DEVICE=eth0   
BOOTPROTO=static   
BROADCAST=XXX.XXX.XXX.255   
IPADDR=XXX.XXX.XXX.XXX   
NETMASK=255.255.255.0   
NETWORK=XXX.XXX.XXX.0   
ONBOOT=yes - Will activate upon system boot   
  
RHEL4/FC3 additions:   
o TYPE=Ethernet   
o HWADDR=XX:XX:XX:XX:XX:XX   
o GATEWAY=XXX.XXX.XXX.XXX   
OR for DHCP client configuration:   
DEVICE=eth0   
ONBOOT=yes   
BOOTPROTO=dhcp   
  
RHEL4/FC3 additions:   
o IPV6INIT=no   
o USERCTL=no   
o PEERDNS=yes   
o TYPE=Ethernet   
o HWADDR=XX:XX:XX:XX:XX:XX   
  
(Used by script /etc/sysconfig/network-scripts/ifup to bring the various network   
interfaces on-line)   
To disable DHCP change BOOTPROTO=dhcp to BOOTPROTO=none   
In order for updated information in any of these files to take effect, one must issue  
the command: service network restart (or: /etc/init.d/network restart)   
  
Changing the host name:  
   
This is a three step process:  
   
1. Issue the command: hostname new-host-name   
2. Change network configuration file: /etc/sysconfig/network   
Edit entry: HOSTNAME=new-host-name   
3. Restart systems which relied on the hostname (or reboot):   
\* Restart network services: service network restart   
(or: /etc/init.d/network restart)   
\* Restart desktop:   
o Bring down system to console mode: init 3   
o Bring up X-Windows: init 5   
One may also want to check the file /etc/hosts for an entry using the system name   
which allows the system to be self aware.   
The hostname may be changed at runtime using the command: sysctl -w   
kernel.hostname="superserver"   
  
  
  
======================================   
NETWORK TUNING SOLARIS:   
======================================   
/sbin/ifconfig hme0:1 inet 10.210.xx.xxx netmask 255.255.0.0 broadcast   
10.210.xxx.xxx   
/\* Virtual Interfaces \*/   
=======================================   
/sbin/ifconfig hme0:1 up   
/\* Bring virtual interface up \*/   
=======================================   
/usr/sbin/ndd -set /dev/hme adv\_100fdx\_cap 1   
/\* Nailling to 100Mbps \*/   
=======================================   
ifconfig eth0 10.1.1.1 netmask 255.255.255.255   
/\* Add an Interface \*/   
=======================================   
ifconfig eth0 mtu 1500   
/\* Change MTU of interface \*/   
=======================================   
ndd -set /dev/ip ip\_addrs\_per\_if 1-8192   
/\* To set more than 256 virtual ip addresses. \*/   
=======================================   
ndd -set /dev/tcp tcp\_recv\_hiwat 65535   
/\* Increase TCP-receivebuffers on Sol2.5.1 systems with 100BaseTx \*/   
=======================================   
ndd -set /dev/tcp tcp\_xmit\_hiwat 65535   
/\* Increase TCP-transmitbuffers on Sol2.5.1 systems with 100BaseTx \*/   
=======================================   
  
======================================   
NETMASK:   
======================================   
Net bits Subnet mask total-addresses   
/20 255.255.240.0 4096   
/21 255.255.248.0 2048   
/22 255.255.252.0 1024   
/23 255.255.254.0 512   
/24 255.255.255.0 256   
/25 255.255.255.128 128   
/26 255.255.255.192 64   
/27 255.255.255.224 32   
/28 255.255.255.240 16   
/29 255.255.255.248 8   
/30 255.255.255.252 4   
Netmask Netmask (binary) CIDR Notes   
255.255.255.255 11111111.11111111.11111111.11111111 /32 Host (single addr)   
255.255.255.254 11111111.11111111.11111111.11111110 /31 Unuseable   
255.255.255.252 11111111.11111111.11111111.11111100 /30 2 useable   
255.255.255.248 11111111.11111111.11111111.11111000 /29 6 useable   
255.255.255.240 11111111.11111111.11111111.11110000 /28 14 useable   
255.255.255.224 11111111.11111111.11111111.11100000 /27 30 useable   
255.255.255.192 11111111.11111111.11111111.11000000 /26 62 useable   
255.255.255.128 11111111.11111111.11111111.10000000 /25 126 useable   
255.255.255.0 11111111.11111111.11111111.00000000 /24 "Class C" 254 useable   
255.255.254.0 11111111.11111111.11111110.00000000 /23 2 Class C's   
255.255.252.0 11111111.11111111.11111100.00000000 /22 4 Class C's   
255.255.248.0 11111111.11111111.11111000.00000000 /21 8 Class C's   
255.255.240.0 11111111.11111111.11110000.00000000 /20 16 Class C's   
255.255.224.0 11111111.11111111.11100000.00000000 /19 32 Class C's   
255.255.192.0 11111111.11111111.11000000.00000000 /18 64 Class C's   
255.255.128.0 11111111.11111111.10000000.00000000 /17 128 Class C's   
255.255.0.0 11111111.11111111.00000000.00000000 /16 "Class B"   
255.254.0.0 11111111.11111110.00000000.00000000 /15 2 Class B's   
255.252.0.0 11111111.11111100.00000000.00000000 /14 4 Class B's   
255.248.0.0 11111111.11111000.00000000.00000000 /13 8 Class B's   
255.240.0.0 11111111.11110000.00000000.00000000 /12 16 Class B's   
255.224.0.0 11111111.11100000.00000000.00000000 /11 32 Class B's   
255.192.0.0 11111111.11000000.00000000.00000000 /10 64 Class B's   
255.128.0.0 11111111.10000000.00000000.00000000 /9 128 Class B's   
255.0.0.0 11111111.00000000.00000000.00000000 /8 "Class A"   
254.0.0.0 11111110.00000000.00000000.00000000 /7   
252.0.0.0 11111100.00000000.00000000.00000000 /6   
248.0.0.0 11111000.00000000.00000000.00000000 /5   
240.0.0.0 11110000.00000000.00000000.00000000 /4   
224.0.0.0 11100000.00000000.00000000.00000000 /3   
192.0.0.0 11000000.00000000.00000000.00000000 /2   
128.0.0.0 10000000.00000000.00000000.00000000 /1   
0.0.0.0 00000000.00000000.00000000.00000000 /0 IP space   
  
  
  
======================================   
OSI MODEL:   
======================================   
Upper Layers   
Layers 7 through 4 comprise the upper layers of the OSI protocol stack. They are more  
geared to the type of application than the lower layers, which are designed to move  
packets, no matter what they contain, from one place to another.   
  
Application Layer 7   
This top layer defines the language and syntax that programs use to communicate with   
other programs. The application layer represents the purpose of communicating in   
the first place. For example, a program in a client workstation uses commands to   
request data from a program in the server. Common functions at this layer are   
opening, closing, reading and writing files, transferring files and e-mail   
messages, executing remote jobs and obtaining directory information about network   
resources.   
  
Presentation Layer 6   
When data are transmitted between different types of computer systems, the   
presentation layer negotiates and manages the way data are represented and encoded.  
For example, it provides a common denominator between ASCII and EBCDIC machines as   
well as between different floating point and binary formats. Sun's XDR and OSI's   
ASN.1 are two protocols used for this purpose. This layer is also used for   
encryption and decryption.   
  
Session Layer 5   
Provides coordination of the communications in an orderly manner. It determines   
one-way or two-way communications and manages the dialog between both parties; for   
example, making sure that the previous request has been fulfilled before the next   
one is sent. It also marks significant parts of the transmitted data with   
checkpoints to allow for fast recovery in the event of a connection failure.   
In practice, this layer is often not used or services within this layer are sometimes  
incorporated into the transport layer.   
  
Transport Layer 4   
This layer is responsible for overall end-to-end validity and integrity of the   
transmission. The lower layers may drop packets, but the transport layer performs a  
sequence check on the data and ensures that if a 12MB file is sent, the full 12MB   
is received.   
"OSI transport services" include layers 1 through 4, collectively responsible for   
delivering a complete message or file from sending to receiving station without   
error.   
  
Lower Layers   
Layers 3 through 1 are responsible for moving packets from the sending station to the  
receiving station.   
  
Network Layer 3   
The network layer establishes the route between the sender and receiver across   
switching points, which are typically routers. The most ubiquitous example of this   
layer is the IP protocol in TCP/IP (see TCP/IP). IPX, SNA and AppleTalk are other   
examples of routable protocols, which means that they include a network address and  
a station address in their addressing system. This layer is also the switching   
function of the dial-up telephone system. If all stations are contained within a   
single network segment, then the routing capability in this layer is not required.   
See layer 3 switch.   
  
Data Link Layer 2   
The data link is responsible for node to node validity and integrity of the   
transmission. The transmitted bits are divided into frames; for example, an   
Ethernet, Token Ring or FDDI frame in local area networks (LANs). Frame relay and   
ATM are also at Layer 2. Layers 1 and 2 are required for every type of   
communications. For more on this layer, see data link protocol.   
  
Physical Layer 1   
The physical layer is responsible for passing bits onto and receiving them from the   
connecting medium. This layer has no understanding of the meaning of the bits, but   
deals with the electrical and mechanical characteristics of the signals and   
signaling methods. For example, it comprises the RTS and CTS signals in an RS-232   
environment, as well as TDM and FDM techniques for multiplexing data on a line.   
SONET also provides layer 1 capability.   
  
  
  
======================================   
OSI Q&A:   
======================================   
What is the difference between layer 2 and layer 3 in the OSI model?   
  
Answer 1:   
The layer2, datalink layer is responsible for moving frames from one hop(node)to the   
next. Whereas in layer3 i.e.., the network layer is responsible for the delivery of  
individual packetsfrom source host to destination host   
  
Answer 2:   
Basically a layer 2 switch operates utilizing Mac addresses in it's caching table to   
quickly pass information from port to port. A layer 3 switch utilizes IP addresses   
to do the same.  
   
While the previous explanation is the "What", for folks in networking the following   
"How" is far more interesting.   
  
Essentially, A layer 2 switch is a multiport bridge. A layer 2 switch will learn   
about MAC addresses connected to each port and pass frames marked for those ports.   
It also knows that if a frame is sent out a port but is looking for the MAC address  
of the port it is connected to it will drop that frame. Whereas a single CPU Bridge  
runs in serial, todays hardware based switches run in parallel, translating to   
extremly fast switching.   
  
Layer 3 switching is a hybrid, as one can imagine, of a router and a switch. There   
are different types of layer 3 switching, route caching and topology-based. In   
route caching the switch requires both a Route Processor (RP) and a Switch Engine   
(SE). The RP must listen to the first packet to determine the destination. At that   
point the Switch Engine makes a shortcut entry in the caching table for the rest of  
the packets to follow. Due to advancement in processing power and drastic   
reductions in the cost of memory, today's higher end layer 3 switches implement a   
topology-based switching which builds a lookup table and and poputlates it with the  
entire network's topology. The database is held in hardware and is referenced there  
to maintain high throughput. It utilizes the longest address match as the layer 3   
destination.   
  
Now when and why would one use a L2 vs L3 vs a router? Simply put, a router will   
generally sit at the gateway between a private and a public network. A router can   
perform NAT whereas an l3 switch cannot (imagine a switch that had the topology   
entries for the ENTIRE Internet!!). In a small very flat network (meaning only one   
private network range for the whole site) a L2 switch to connect all the servers   
and clients to the internet is probably going to suffice. Larger networks, or those  
with the need to contain broadcast traffic or those utilizing VOIP, a multi network  
approach utilizing VLANs is appropriate, and when one is utilizing VLANs, L3   
switches are a natural fit. While a router on a stick scenario can work, it can   
quickly overtax a router if there is any significant inter-vlan traffic since the   
router must make complicated routing decisions for every packet that it receives.   
  
  
  
======================================   
What's the difference between a hub, a switch, and a router?   
======================================   
A hub is typically the least expensive, least intelligent, and least complicated of   
the three. Its job is very simple: anything that comes in one port is sent out to   
the others. That's it. Every computer connected to the hub "sees" everything that   
every other computer on the hub sees. The hub itself is blissfully ignorant of the   
data being transmitted. For years, simple hubs have been quick and easy ways to   
connect computers in small networks.   
  
A switch does essentially what a hub does but more efficiently. By paying attention   
to the traffic that comes across it, it can "learn" where particular addresses are.  
For example, if it sees traffic from machine A coming in on port 2, it now knows   
that machine A is connected to that port and that traffic to machine A needs to   
only be sent to that port and not any of the others. The net result of using a   
switch over a hub is that most of the network traffic only goes where it needs to   
rather than to every port. On busy networks this can make the network significantly  
faster.   
  
A router is the smartest and most complicated of the bunch. Routers come in all   
shapes and sizes from the small four-port broadband routers that are very popular   
right now to the large industrial strength devices that drive the internet itself.   
A simple way to think of a router is as a computer that can be programmed to   
understand, possibly manipulate, and route the data its being asked to handle. For   
example, broadband routers include the ability to "hide" computers behind a type of  
firewall which involves slightly modifying the packets of network traffic as they   
traverse the device. All routers include some kind of user interface for   
configuring how the router will treat traffic. The really large routers include the  
equivalent of a full-blown programming language to describe how they should operate  
as well as the ability to communicate with other routers to describe or determine   
the best way to get network traffic from point A to point B.   
  
  
  
======================================   
SWITCHING /////////////////////////   
======================================   
  
  
======================================   
CREATE VLAN IOS:   
======================================   
This example shows how to create an Ethernet VLAN in global configuration mode and   
verify the configuration:   
Router# configure terminal   
Router(config)# vlan 3   
Router(config-vlan)# end   
Router# show vlan id 3   
VLAN Name Status Ports   
---- -------------------------------- --------- -------------------------------   
3 VLAN0003 active   
VLAN Type SAID MTU Parent RingNo BridgeNo Stp BrdgMode Trans1 Trans2   
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------   
3 enet 100003 1500 - - - - - 0 0   
Primary Secondary Type Interfaces   
------- --------- ----------------- ------------------------------------------   
This example shows how to create an Ethernet VLAN in VLAN database mode:   
Router# vlan database   
Router(vlan)# vlan 3   
VLAN 3 added:   
Name: VLAN0003   
Router(vlan)# exit   
APPLY completed.   
Exiting....   
This example shows how to verify the configuration:   
Router# show vlan name VLAN0003   
VLAN Name Status Ports   
---- -------------------------------- --------- ---------------------   
3 VLAN0003 active   
VLAN Type SAID MTU Parent RingNo BridgeNo Stp Trans1 Trans2   
---- ----- ---------- ----- ------ ------ -------- ---- ------ ------   
3 enet 100003 1500 - - - - 0 0   
Router#   
This example shows how to map 802.1Q VLAN 1003 to ISL VLAN 200:   
Router# configure terminal   
Router(config)# vlan mapping dot1q 1003 isl 200   
Router(config)# end   
Router#   
This example shows how to verify the configuration:   
Router# show vlan   
<...output truncated...>   
802.1Q Trunk Remapped VLANs:   
802.1Q VLAN ISL VLAN   
----------- -----------   
1003 200   
  
  
  
======================================   
ROUTING /////////////////////////   
======================================   
  
  
======================================   
Cisco Router Configuration Commands:   
======================================   
Set a console password to cisco   
Router(config)#line con 0   
Router(config-line)#login   
Router(config-line)#password cisco   
======================================   
Set a telnet password   
Router(config)#line vty 0 4   
Router(config-line)#login   
Router(config-line)#password cisco   
======================================   
Stop console timing out   
Router(config)#line con 0   
Router(config-line)#exec-timeout 0 0   
======================================   
Set the enable password to cisco   
Router(config)#enable password cisco   
======================================   
Set the enable secret password to peter.   
This password overrides the enable password   
and is encypted within the config file   
Router(config)#enable secret peter   
======================================   
Enable an interface   
Router(config-if)#no shutdown   
======================================   
To disable an interface   
Router(config-if)#shutdown   
======================================   
Set the clock rate for a router with a DCE cable to 64K   
Router(config-if)clock rate 64000   
======================================   
Set a logical bandwidth assignment of 64K to the serial interface   
Router(config-if)bandwidth 64   
Note that the zeroes are not missing   
======================================   
To add an IP address to a interface   
Router(config-if)#ip addr 10.1.1.1 255.255.255.0   
======================================   
To enable RIP on all 172.16.x.y interfaces   
Router(config)#router rip   
Router(config-router)#network 172.16.0.0   
Disable RIP Router(config)#no router rip   
======================================   
To enable IRGP with a AS of 200, to all interfaces   
Router(config)#router igrp 200   
Router(config-router)#network 172.16.0.0   
Disable IGRP Router(config)#no router igrp 200   
======================================   
Static route the remote network is 172.16.1.0,   
with a mask of 255.255.255.0, the next  
hop is 172.16.2.1, at a cost of 5 hops   
Router(config)#ip route 172.16.1.0 255.255.255.0 172.16.2.1 5   
======================================   
Disable CDP for the whole router   
Router(config)#no cdp run   
======================================   
Enable CDP for he whole router   
Router(config)#cdp run   
======================================   
Disable CDP on an interface   
Router(config-if)#no cdp enable   
======================================

======================================   
Cisco Router Show Commands:   
======================================   
View version information   
show version   
======================================   
View current configuration (DRAM)   
show running-config   
======================================   
View startup configuration (NVRAM)   
show startup-config   
======================================   
Show IOS file and flash space   
show flash   
======================================   
Shows all logs that the router has in its memory   
show log   
======================================   
View the interface status of interface e0   
show interface e0   
======================================   
Overview all interfaces on the router   
show ip interfaces brief   
======================================   
View type of serial cable on s0   
show controllers 0 (note the space between the 's' and the '0')   
======================================   
Display a summary of connected cdp devices   
show cdp neighbor   
======================================   
Display detailed information on all devices   
show cdp entry \*   
======================================   
Display current routing protocols   
show ip protocols   
======================================   
Display IP routing table   
show ip route   
======================================   
Display access lists, this includes the number of displayed matches   
show access-lists   
======================================   
Check the router can see the ISDN switch   
show isdn status   
======================================   
Check a Frame Relay PVC connections   
show frame-relay pvc   
======================================   
show lmi traffic stats   
show frame-relay lmi   
======================================   
Display the frame inverse ARP table   
show frame-relay map   
======================================   
======================================   
Cisco Router Basic Operations   
======================================   
Enable   
Enter privileged mode   
======================================   
Return to user mode from privileged   
disable   
======================================   
Exit Router   
Logout or exit or quit   
======================================   
Recall last command   
up arrow or <Ctrl-P>   
======================================   
Recall next command   
down arrow or <Ctrl-N>   
======================================   
Suspend or abort   
<Shift> and <Ctrl> and 6 then x   
======================================   
Refresh screen output   
<Ctrl-R>   
======================================   
Complete Command   
TAB   
======================================   
  
  
  
======================================   
Cisco Router Copy Commands:   
======================================   
Save the current configuration from DRAM to NVRAM   
copy running-config startup-config   
======================================   
Merge NVRAM configuration to DRAM   
copy startup-config running-config   
======================================   
Copy DRAM configuration to a TFTP server   
copy runing-config tftp   
======================================   
Merge TFTP configuration with current router configuration held in DRAM   
copy tftp runing-config   
======================================   
Backup the IOS onto a TFTP server   
copy flash tftp   
======================================   
Upgrade the router IOS from a TFTP server   
copy tftp flash   
======================================   
  
  
  
======================================   
Cisco Router Debug Commands:   
======================================   
Enable debug for RIP   
debug ip rip   
======================================   
Enable summary IGRP debug information   
debug ip igrp events   
======================================   
Enable detailed IGRP debug information   
debug ip igrp transactions   
======================================   
Debug IPX RIP   
debug ipx routing activity   
======================================   
Debug IPX SAP   
debug IPX SAP   
======================================   
Enable debug for CHAP or PAP   
debug ppp authentication   
======================================   
Switch all debugging off   
no debug all   
undebug all   
======================================   
  
  
  
======================================   
SCRIPTS ////////////////////////   
======================================   
  
  
======================================   
Bash Cleanup Script:   
======================================   
All the files below go in the same directory   
"README.1st", "rotatelog", rotatelog.rc", and "rotatelog.lsm"  
   
# README.1st FILE  
#####################################   
# rotatelog Ver 0.2   
####################################   
1 Introduction   
1.1 Rotation of log files is a mandatory task of every sys-   
admin, failing which they grow beyond proportions. Many   
distros cater for utilities like logrotate, which are   
usually fired as a cron process. These are essentially   
shell scripts, and take care of these routine chores.   
1.2 I for one have always been advocating manual admining,   
since you know what is going on. I have been doing rot-   
ation of log files manually through scripts for many   
years now without problems.   
1.3 Since location of log files are distro specific, there   
is a need to evolve a method which will work for all.   
  
2 Installation   
2.1 Copy this script to a suitable place. The recommended   
place is at /usr/sbin (so that no user can access it).   
It has an in-built check for superuser privileges, so,   
placing this in /usr/local/bin will do just as well.   
2.2 Documentation ? None, except for this file, which may be   
removed, once you are sailing. The script itself is also   
heavily commented.   
  
3 rotatelog.rc   
3.1 This is merely a bash script. This rotates files in an   
interactive mode. At the heart of the process is an rc   
file which contains the list of all log files which are   
known to grow with time. The log files may be anywhere   
on the system. A sample rc file is placed below:   
####################################   
# Sample rotatelog.rc file.All lines   
# with # are omitted. All filenames   
# with full path, to begin on Col 1.   
# No line gaps permitted in between.   
####################################   
/var/log/messages   
/var/log/syslog   
/var/log/wtmp   
/var/log/debug   
/home/bish/mail/.procmail.log   
/var/log/boa/access\_log   
/var/log/boa/error\_log   
  
3.2 By default, this file is kept at /var/log/oldlogs/ dir,   
though it can be kept anywhere as per the location that   
is specified on top of the script. With this strategy,   
the reach of the program is virtually endless, and any   
log file anywhere can be added to the list.   
  
4 Running rotatelog   
4.1 This script must be run with root privileges. This is   
a sys-adm function. The script runs with command line   
options:   
--help or -h ... for help   
-i ... For information on the file sizes of the log   
files specified in the rc file.   
-e ... To execute a rotation selectively. Sizes of   
all log files are displayed. It is then pos-   
sible to selectively choose the file(s) for   
rotation.   
5. Features of rotatelog   
5.1 None of the log files are removed. They are moved to a   
selected directory ( default being /var/log/oldlogs )   
and then gzipped. The location of the log archive is   
also editable within the script.   
5.2 All new log files are re-initialised thereafter.   
5.3 Optionally, it is possible to send a mail to "root" as   
and when a log file is rotated and archived. This too   
is user selectable.   
6 A sample session   
-----------------<sample session>---------------------   
aedes:~#rotatelog -e   
rotatelog Version : 0.2   
1] 45056 /var/log/messages   
2] 8192 /var/log/syslog   
3] 106496 /var/log/wtmp   
4] 28672 /var/log/debug   
5] 4096 /home/bish/mail/.procmail.log   
6] 106496 /var/log/boa/access\_log   
7] 163840 /var/log/boa/error\_log   
Which logs to rotate [1 - 7] ?   
Otherwise enter [0] ... to abort   
Enter numbers with with spaces in between : 6 7   
Rotating ... /var/log/boa/access\_log   
Rotating ... /var/log/boa/error\_log   
-- Mail to root sent --   
aedes:~#mail   
Mail version 8.1 6/6/93. Type ? for help.   
/var/spool/mail/root: 1 message 1 new   
& 1   
Message 1:   
From root Sun Nov 4 16:24:07 2001   
Date: Sun, 4 Nov 2001 16:24:07 +0530   
From: rotatelog@aedes   
To: root@aedes   
Subject: rotatelog notice   
Sun Nov 4 16:24:07 IST 2001   
/var/log/oldlogs/access\_log.011104.gz   
/var/log/oldlogs/error\_log.011104.gz   
& quit   
Saved 1 message in mbox   
aedes:~#exit   
--------------------</sample session>------------------   
6.1 Please note the following from the sample session   
o All log files in the rc files have been shown   
o It was possible to selectively rotate only two of the   
files (in this example the log files of boa http ser-   
ver), by entering 6 and 7 seperated by a space. It is   
possible to rotate multiple log files selectively.   
o The files have been archived with datestamp. A middle   
6-digit number is added, which is the yymmdd stamp   
for the day. It is assumed that no file would need to   
be rotated twice in a day !   
  
7 Pre-requisites   
7.1 For mailing to root, sendmail binary is used directly,   
so a properly configured MTA is expected. No recourse   
is taken for mailing through any MUA.   
7.2 Ensure that you go through the script, and edit the   
configurable section on the top for your system.   
7.3 There is NO need to create a directory for the archived   
log files, nor any need to create a rc file. The first   
time this script is run, it checks for the presence of   
the needed directories and rc file, if not found, crea-   
tes one. You may edit the rc file subsequently with   
root privileges.   
  
8 Bugs and Bunnies   
8.1 I have been using this script (in a less refined format   
since 1996). However, if you face any problems be free   
to contact me. There is only one bunny (I expect). The   
number of log files is limited to about 20 since after   
that, the top ones would scroll off the screen. I have   
never had the opportunity to cross this limit. In case   
the need is felt, it would be necessary to modify the   
chk\_size routine to be passed through a pager ... can do   
it if asked ... till then, let things lie as they are.   
  
9 Kudos and Brickbats   
9.1 This script is released under GNU/ GPL licence. You are   
free to use and distribute this without any encumberan-   
ces. Ofcourse as per the protection of the GNU licence,   
no guns can be pointed at me if things go wrong at any   
time because of this script ;-)   
9.2 I would love to have any extensions or alterations made   
to this script ... all kudos and brickbats should be   
directed at:   
USM Bish [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
04 Nov 2001   
######################################   
######################################   
  
  
  
# ROTATE LOG SCRIPT   
######################################   
#!/bin/bash   
######################################  
  
#   
# Shell program to rotate log files in /var/spool and other dirs   
# anywhere on the system. Log files to be checked are to be inc-   
# luded in a seperate rc file. See docs for format of file.   
#   
# Copyright 2001, USM Bish, [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
#   
# This program is free software; you can redistribute it and/ or   
# modify it under the terms of the GNU General Public License as   
# published by the Free Software Foundation; either version 2 of   
# the License, or (at your option) any later version.   
#   
# This program is distributed in the hope that it will be useful   
# but WITHOUT ANY WARRANTY; without even the implied warranty of   
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the   
# GNU General Public License for more details.   
#   
# Description: A distro-independent method of rotating log files   
# in /var/spool without necessity for cron processes.   
#   
# NOTE: You must be the superuser to run this script.   
#   
# Usage:   
#   
# rotatelog [ -h | --help ] [-e] [-i]   
#   
# Options:   
#   
# -h, --help Display this help message and exit.   
# -e E]xecute rotation   
# -i I]nfo on present log file sizes   
#   
#   
# Revisions:   
#   
# Nov/03/2001 File created .... ver 0.1   
# Nov/04/2001 Mail to root added .... ver 0.2   
#   
################################  
  
################################   
# Editable variables   
################################   
#### Where to archive the old logs ?   
OLDLOGS=/var/log/oldlogs   
#### Where is the rotatelogrc file ?   
RC=$OLDLOGS/rotatelog.rc   
#### Send mail to root ? [ yes / no ]   
SENDMAIL="yes"   
#SENDMAIL="no"   
#################################   
No editing below this   
#################################   
PROGNAME=$(basename $0)   
VERSION="0.2"   
TEMP\_FILE1=/tmp/${PROGNAME}.$$.1   
TEMP\_FILE2=/tmp/${PROGNAME}.$$.2   
TODAY=$(date +%y%m%d) # YYMMDD for convenient sorting   
################################   
# Functions   
################################  
function send\_mail   
{   
#### Send mail to root   
echo "#!/bin/sh" > $TEMP\_FILE1   
echo "sendmail -t << -EndOfMail-" >> $TEMP\_FILE1   
echo "From: rotatelog" >> $TEMP\_FILE1   
echo "To: root" >> $TEMP\_FILE1   
echo "Subject: rotatelog notice" >> $TEMP\_FILE1   
echo "" >> $TEMP\_FILE1   
date >> $TEMP\_FILE1   
echo "" >> $TEMP\_FILE1   
cat $TEMP\_FILE2 >> $TEMP\_FILE1   
echo "" >> $TEMP\_FILE2   
echo "-EndOfMail-" >> $TEMP\_FILE1   
chmod +x $TEMP\_FILE1   
$TEMP\_FILE1   
echo ""   
echo "-- Mail to root sent --"   
echo ""   
}   
function chk\_size   
{   
CNT=0   
for i in `cat $RC | grep -v "#"`; do   
CNT=$((CNT+1))   
echo -en $CNT"]\t"   
du -b $i   
done   
}   
function rotate\_log   
{   
#### Create the backup and zip it   
BASENAME=`basename $TARGET`   
cp $TARGET $OLDLOGS/$BASENAME.$TODAY   
gzip -9 $OLDLOGS/$BASENAME.$TODAY   
echo $OLDLOGS/$BASENAME.$TODAY.gz >> $TEMP\_FILE2   
#### Now zap the space occupying hogs   
cat /dev/null > $TARGET   
chmod 666 $TARGET   
}   
function chk\_rc   
{   
if ! [ -s $RC ]; then   
echo "rc file NOT found ... "$RC   
echo -en "Create one ? [y/n] : "   
read YN   
case $YN in   
Y|y) ## Create a skeleton RC file   
cat << -EndOfRC- > $RC   
##########################  
# Sample rotatelog.rc file.All lines   
# with # are omitted. All filenames   
# with full path, to begin on Col 1.   
# No line gaps permitted in between.   
##########################  
/var/log/messages   
/var/log/syslog   
/var/log/wtmp   
/var/log/debug   
-EndOfRC-   
echo   
echo $RC" has been created"   
echo "Add to this file if more log files are to"   
echo "be included ... Press [Enter] to continue"   
echo   
read   
clear   
;;   
\*) # Anything else entered   
echo "Cannot proceed without a rc file"   
term\_exit   
esac   
fi   
}   
  
function chk\_oldlogs   
{   
if ! [ -d $OLDLOGS ]; then   
echo -en "\tFirst time run ...\n\n"   
mkdir $OLDLOGS   
chmod 755 $OLDLOGS   
fi   
}   
function chk\_root   
{   
USR=`whoami`   
if ! [ "$USR" = "root" ]; then   
echo $PROGNAME" [Version : "$VERSION"]"   
echo "Root privileges needed ... "   
term\_exit   
fi   
}   
function clean\_up   
{   
rm -f $TEMP\_FILE1   
rm -f $TEMP\_FILE2   
}   
  
function graceful\_exit   
{   
clean\_up   
exit   
}   
  
function error\_exit   
{   
echo "${PROGNAME}: ${1:-"Unknown Error"}" >&2   
clean\_up   
exit 1   
}   
function term\_exit   
{   
echo "${PROGNAME}: Terminated"   
clean\_up   
exit   
}   
  
function int\_exit   
{   
echo "${PROGNAME}: Aborted by user"   
clean\_up   
exit   
}   
  
function usage   
{   
echo "Usage: ${PROGNAME} [-h | --help] [-e] [-i]"   
}   
  
function helptext   
{   
local tab=$(echo -en "\t\t")   
cat <<- -EOF-   
${PROGNAME} ver. ${VERSION}   
This is a program to rotate log files in /var/spool or   
any other directory on the system, as specified in the   
rc file : $RC   
$(usage)   
Options:   
-h, --help Display this help message and exit.   
-e E]xecute rotation   
-i I]nfo on present log file sizes   
NOTE: You must be the superuser to run this script.   
-EOF-   
}   
#####################################  
# Program starts here   
#####################################  
# Trap TERM, HUP, and INT signals and properly exit   
trap term\_exit TERM HUP   
trap int\_exit INT   
if [ "$1" = "" ]; then   
usage   
graceful\_exit   
fi   
if [ "$1" = "--help" ]; then   
helptext   
graceful\_exit   
fi   
chk\_root   
chk\_oldlogs   
chk\_rc   
# Process arguments   
while getopts ":hei" opt; do   
case $opt in   
e ) echo   
echo $PROGNAME" Version : "$VERSION   
echo   
chk\_size   
echo   
echo "Which logs to rotate [1 - $CNT] ?"   
echo "Otherwise enter [0] ... to abort."   
echo -en "Enter numbers with with spaces in between : "   
read NOS   
TNOS=`echo $NOS | tr -d [:alpha:]`   
NOS=$TNOS   
if [ "$NOS" = "" ]; then   
echo "Invalid option"   
term\_exit   
fi   
if [ "$NOS" = "0" ]; then   
term\_exit   
fi   
NNNOS=`echo $NOS | awk '{print $1}'`   
if [ "$NNNOS" -gt "$CNT" ]; then   
echo   
echo "Number not in menu : "$NOS   
term\_exit   
fi   
NCNT=0   
clean\_up   
for i in `echo $NOS`; do   
NCNT=0   
for j in `cat $RC | grep -v "#"`; do   
NCNT=$((NCNT+1))   
if [ "$NCNT" = "$i" ]; then   
TARGET=$j   
echo -en "\nRotating ... $TARGET\n"   
rotate\_log   
fi   
done   
done   
if [ "$SENDMAIL" = "yes" ]; then   
send\_mail   
fi   
;;   
i ) echo   
echo $PROGNAME" Version : "$VERSION   
echo   
echo "Info on present sizes of logs in bytes :"   
echo   
chk\_size   
echo   
echo "Do: "$PROGNAME" -e ... to rotate these logs"   
echo   
;;   
h ) helptext   
graceful\_exit   
;;   
\* ) usage   
exit 1   
esac   
done   
graceful\_exit   
###########################  
# Everything below this is ignored   
###########################  
  
  
  
# ROTATELOG.RC FILE   
####################################   
  
  
####################################   
# Sample rotatelog.rc file.All lines   
# with # are omitted. All filenames   
# with full path, to begin on Col 1.   
# No line gaps permitted in between.   
####################################   
/var/log/messages   
/var/log/syslog   
/var/log/wtmp   
/var/log/debug   
/home/bish/mail/.procmail.log   
/var/log/boa/access\_log   
/var/log/boa/error\_log   
"rotatelog.lsm"   
Begin3   
Title: rotatelog   
Version: 0.2   
Entered-date: 04 Nov 2001   
Description: Shell script manage log files in /var/log and other   
log files anywhere on the system   
Keywords: sysadmin, logrotate, rotate log files   
Author: [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
(USM Bish)   
Maintained-by: [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
(USM Bish)   
Primary-site: http://geocities.com/usmbish/scripts.html   
Original-site: http://geocities.com/usmbish/scripts.html   
Platforms: Linux   
Copying-policy: GPL   
End   
  
  
# rotatelog.lsm   
#################################   
  
Begin3   
Title: rotatelog   
Version: 0.2   
Entered-date: 04 Nov 2001   
Description: Shell script manage log files in /var/log and other   
log files anywhere on the system   
Keywords: sysadmin, logrotate, rotate log files   
Author: [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
(USM Bish)   
Maintained-by: [bish@nde.vsnl.net.in](mailto:bish@nde.vsnl.net.in)   
(USM Bish)   
Primary-site: http://geocities.com/usmbish/scripts.html   
Original-site: http://geocities.com/usmbish/scripts.html   
Platforms: Linux   
Copying-policy: GPL   
End

======================================   
 Bash script to validate RPM files:   
======================================   
   
   
   
 #!/bin/bash   
 # rpm-check.sh   
   
 # Queries an rpm file for description, listing,   
 #+ and whether it can be installed.   
 # Saves output to a file.   
 #   
 # This script illustrates using a code block.   
   
 SUCCESS=0   
 E\_NOARGS=65   
   
 if [ -z "$1" ]   
 then   
 echo "Usage: `basename $0` rpm-file"   
 exit $E\_NOARGS   
 fi   
   
 { # Begin code block.   
 echo   
 echo "Archive Description:"   
 rpm -qpi $1 # Query description.   
 echo   
 echo "Archive Listing:"   
 rpm -qpl $1 # Query listing.   
 echo   
 rpm -i --test $1 # Query whether rpm file can be installed.   
 if [ "$?" -eq $SUCCESS ]   
 then   
 echo "$1 can be installed."   
 else   
 echo "$1 cannot be installed."   
 fi   
 echo # End code block.   
 } > "$1.test" # Redirects output of everything in block to file.   
   
 echo "Results of rpm test in file $1.test"   
   
 # See rpm man page for explanation of options.   
   
 exit 0   
   
   
   
 ======================================   
 Crontab:   
 ======================================   
   
 Creating a crontab file   
 ======================================   
 crontab -e   
 ======================================   
   
 Crontab syntax   
 ======================================   
 \* \* \* \* \* command to be executed   
 - - - - -   
 | | | | |   
 | | | | +----- day of week (0 - 6) (Sunday=0)   
 | | | +------- month (1 - 12)   
 | | +--------- day of month (1 - 31)   
 | +----------- hour (0 - 23)   
 +------------- min (0 - 59)   
 ======================================   
   
 Crontab examples   
 ======================================   
 \* \* \* \* \* <command> #Runs every minute   
 30 \* \* \* \* <command> #Runs at 30 minutes past the hour   
 45 6 \* \* \* <command> #Runs at 6:45 am every day   
 45 18 \* \* \* <command> #Runs at 6:45 pm every day   
 00 1 \* \* 0 <command> #Runs at 1:00 am every Sunday   
 00 1 \* \* 7 <command> #Runs at 1:00 am every Sunday   
 00 1 \* \* Sun <command> #Runs at 1:00 am every Sunday   
 30 8 1 \* \* <command> #Runs at 8:30 am on the first day of every month   
 00 0-23/2 02 07 \* <command> #Runs every other hour on the 2nd of July   
 ======================================   
   
 Special Stirngs   
 ======================================   
 @reboot <command> #Runs at boot   
 @yearly <command> #Runs once a year [0 0 1 1 \*]   
 @annually <command> #Runs once a year [0 0 1 1 \*]   
 @monthly <command> #Runs once a month [0 0 1 \* \*]   
 @weekly <command> #Runs once a week [0 0 \* \* 0]   
 @daily <command> #Runs once a day [0 0 \* \* \*]   
 @midnight <command> #Runs once a day [0 0 \* \* \*]   
 @hourly <command> #Runs once an hour [0 \* \* \* \*]   
 ======================================   
   
 Multiple commands   
   
 @daily <command\_01> && <command\_02>   
 ======================================   
   
 Disabling email notifications   
   
 By default a cron job will send an email to the user account executing the cronjob.   
 If this is not needed put the following command at the end of the cron job line:   
 >/dev/null 2>&1   
 ======================================   
   
 Specifying a crontab file to use   
   
 crontab -u <username> <crontab file>   
 crontab -u tux ~/crontab   
 ======================================   
   
 Removing a crontab file   
   
 crontab -r   
 ======================================   
   
   
   
   
   
   
 ======================================   
 JUMPSTART ////////////////////////   
 ======================================   
   
 # mkdir /jumpstart/image   
 # mkdir /jumpstart/config   
 # mkdir /jumpstart/share   
   
 # lofiadm -a /var/tmp/Solaris10\_u5\_1108.iso   
 /dev/lofi/1   
 # lofiadm /dev/lofi/1   
 /var/tmp/Solaris10\_u5\_1108.iso   
   
 # svcadm disable volfs   
 # mkdir -p /cdrom/cdrom0   
   
 # mount -F hsfs -o ro /dev/lofi/1 /cdrom/cdrom0   
   
 # cd /cdrom/cdrom0/Solaris\_10/Tools   
 # ./setup\_install\_server /jumpstart/image   
 Verifying target directory...   
 Calculating the required disk space for the Solaris\_11 product   
 ... output skipped ...   
   
 # cd /   
 # umount /cdrom/cdrom0   
 # lofiadm -d /dev/lofi/1   
 # lofiadm   
 Block Device File   
   
 # cd /jumpstart/image/Solaris\_10/Misc/jumpstart\_sample   
 # cp ./check /jumpstart/config   
   
 # cp /etc/dfs/dfstab /etc/dfs/dfstab.org   
   
 # vi /etc/dfs/dfstab   
 +-------------------   
 | share -F nfs -o ro,anon=0 /jumpstart/config   
 | share -F nfs -o ro,anon=0 /jumpstart/image   
 | share -F nfs -o ro,anon=0 /jumpstart/share   
   
 # vi /etc/dfs/dfstab   
 +-------------------   
 | share -F nfs -o ro,anon=0 /jumpstart   
   
 # svcadm enable nfs/server   
 # shareall   
   
 # vi /jumpstart/config/sysidcfg   
 +------------------------------   
 | system\_locale=en\_US   
 | timezone=MET   
 | name\_service=NONE   
 | terminal=dtterm   
 | timeserver=localhost   
 | root\_password="WybF.D5GwZnz2"   
 | network\_interface=primary { netmask=255.0.0.0 protocol\_ipv6=no   
 default\_route=127.0.0.1}   
 | security\_policy=NONE   
 | nfs4\_domain=dynamic   
   
 # vi /jumpstart/config/sun4u\_profile   
 +-----------------------------------   
 | install\_type initial\_install   
 | system\_type standalone   
 | partitioning explicit   
 | filesys any 1024 /   
 | filesys any 1024 /usr   
 | filesys any 1024 /var   
 | filesys any 1024 /opt   
 | filesys any 1024 /export/home   
 | filesys any 256 swap   
 | cluster SUNWCreq   
 | package SUNWman   
 | package SUNWbash   
 | package SUNWless   
   
 # cd /jumpstart/config   
 # vi ./rules   
 +-----------   
 | karch sun4u - sun4u\_profile -   
   
 # ./check   
   
 # vi /etc/hosts   
 +--------------   
 | 10.0.0.2 pino   
   
 # cd /jumpstart/image/Solaris\_10/Tools   
 # ./add\_install\_client \   
 > -e 8:0:20:0:0:02 \   
 > -i 10.0.0.2 \   
 > -s tommie:/jumpstart/image \   
 > -c tommie:/jumpstart/config \   
 > -p tommie:/jumpstart/config \   
 > pino \   
 > sun4u   
   
 # svcadm enable rarp   
   
 # inetconv   
   
 # init 0   
 ok boot net - install   
   
 Create a finish script   
   
 # vi /jumpstart/config/sun4u\_after   
 +---------------------------------   
 | {   
 | mkdir /a/server   
 | mount -F nfs -o ro 10.0.0.1:/jumpstart/share /a/server   
 |   
 | cp /a/server/crontab.root /a/var/spool/cron/crontabs/root   
 | cp /a/server/hosts.header /a/hosts   
 |   
 | HOSTNAME=`cat /etc/nodename`   
 | regel=`grep $HOSTNAME /a/server/hosts.org`   
 | echo "$regel loghost ." >> /a/hosts   
 | grep -v $HOSTNAME /a/server/hosts.org >> /a/hosts   
 |   
 | mv /a/hosts /a/etc/hosts   
 | | umount /a/server   
 | rmdir /a/server   
 |   
 | touch /a/noautoshutdown   
 | touch /a/etc/.NFS4inst\_state.domain   
 | } > /a/server.log 2> /a/server.errlog   
   
 # vi /jumpstart/share/hosts.header   
 +---------------------------------   
 | #   
 | # Internet host table   
 | #   
   
 # vi /jumpstart/share/hosts.org   
 +------------------------------   
 | 10.0.0.1 tommie   
 | 10.0.0.2 pino # crontab -l > /jumpstart/share/crontab.root   
   
 Update the rules file   
   
 # vi rules   
 +-----------   
 | karch sun4u - sun4u\_profile sun4u\_after   
   
 # ./check   
   
   
   
   
 ======================================   
 KICKSTART ////////////////////////   
 ======================================   
   
   
 Before You Start   
   
 A Kickstart install involves three participants: a target machine uses a config file   
 to set system parameters and determine what RPMs to pull from the installation   
 media. (The config file may have any name; this article will refer to it as   
 ks.cfg.)   
   
 There are several ways to connect those pieces: the target machine can fetch the RPMs  
 from a local disk, NFS server, FTP server, and so on. The config file can come from  
 the aforementioned places or from the boot media, and it may exist in a different   
 place than the installation media.   
   
 Such flexibility makes it difficult to explain a "typical" Kickstart process in   
 detail. This article demonstrates just one method, using a web server to host the   
 install media and config file. This is likely the easiest and least intrusive   
 method to experiment with Kickstart. It should also scale as your Kickstart   
 experiment matures into a formal infrastructure.   
   
 To that end, the setup described in this article requires:   
   
 \* The Fedora install files, which you'll copy to the web server's file system.   
 \* A target machine on which you will install Fedora. Using virtual hardware, such as   
 VMware or Bochs, will simplify your experiment.   
  
 \* Bootable media that matches the version of Fedora you plan to install. Choose from   
 install CD 1, diskettes (images/bootdisk.img and images/drvnet.img from the install  
 media), or a bootable CD made from images/boot.iso.  
   
 \* A source machine to host the install files and Kickstart config, and run the web   
 server.   
   
 Some of these require additional explanation and I'll describe them in turn.   
 The Source Machine: Setting up the Install Tree  
   
 The target machine will fetch its install files and ks.cfg from a web server running   
 on the source machine. The source machine needn't run Linux, but it must have   
 roughly 2.2G disk space available. The web server must listen on port 80 due to a   
 limitation in Kickstart's HTTP code.   
   
 Create a directory FC1-install under the document root and populate it with the   
 Fedora directory from the install media. Use your preferred download tool (say,   
 wget) to grab the tree from a Fedora mirror site or copy the contents from the   
 install CDs or ISOs. Be sure to maintain the directory structure in this latter   
 case. There are myriad ways to do this, such as:   
   
 $ cd /mnt/cdrom   
 $ cp -a Fedora /...docroot.../FC1-install   
   
 Creating the Kickstart Config File, ks.cfg   
   
 ks.cfg makes unattended installs possible. It holds canned responses to the questions  
 posed during an interactive install. The examples assume you've saved this file   
 under the web server's document root as kickstart/ks.cfg.   
   
 There are several ways to create ks.cfg. (I did warn you that Kickstart was   
 flexible.) If you're plotting a clone farm, build one machine to your specs and use  
 /root/anaconda-ks.cfg on that host as a starting point for the others.   
   
 Barring that, use the redhat-config-kickstart GUI (from the redhat-config-kickstart   
 package). This tool doesn't support LVM for disk layout, but is a valuable learning  
 tool nonetheless. You can hand-edit the generated ks.cfg to use LVM (described   
 below).   
   
 You can also create or edit ks.cfg using any text editor, provided you know the   
 directives. Here's a walk through the directives in the sample ks.cfg.   
 You probably already have the redhat-config-language, hwdata, and tzdata RPMs   
 installed already. They are not required, but include files that simplify   
 hand-editing ks.cfg.   
 Installation Type   
   
 The first entries are the installation type and source.   
   
 install   
 url --url http://kickstart-server/FC1-install   
   
 The type may be install or upgrade. The url directive specifies an HTTP installation   
 and indicates the URL of the install media. (The directory Fedora, from the install  
 media, must be a subdirectory of the URI part of the URL.) Other installation   
 sources include cdrom for swapping CDs or DVDs, nfs for mounting the install media   
 from an NFS share, and the self-explanatory ftp.  
   
 Languages and Input   
 lang and mouse indicate the language and mouse type, respectively, to use during the   
 installation.   
   
 lang en\_US.UTF-8   
 mouse generic3ps/2   
   
 The sample ks.cfg uses U.S. English with the Unicode (UTF-8) character set, and a   
 generic PS2 mouse with three buttons.   
   
 Refer to /usr/share/redhat-config-language/locale-list for the list of valid   
 languages.   
   
 The values of lang and mouse don't matter for unattended installations.   
 langsupport and keyboard set the runtime (installed) language support and keyboard   
 type, respectively.   
   
 langsupport --default en\_US.UTF-8 en\_US.UTF-8  
 keyboard us   
   
 Specify a single language (en\_US) or multiple languages with a default (--default   
 en\_US en\_UK). Specifying just the default (--default en\_US) installs support for   
 all languages.   
 Video   
   
 For a workstation build you'll likely want to configure your video card and monitor,   
 using xconfig.   
   
 xconfig --card "VMWare" --videoram 16384 --hsync 31.5-37.9  
 --vsync 50-70 --resolution 800x600 --depth 16   
   
 (We've split the above line for readability; it should be a single line in ks.cfg..)   
 xconfig takes the card's name (listed in /usr/share/hwdata/Cards) and video RAM in   
 kilobytes. The remaining parameters specify the monitor's horizontal and vertical   
 sync rates, resolution, and color depth in bits.   
   
 Use the skipx directive to skip this step (say, for headless servers). You can   
 manually configure X later.   
   
 Networking   
   
 The network directive sets the target host's runtime network configuration. This may   
 be different than the build-time IP. For example, you may use separate networks to   
 build (DHCP-enabled) and deploy machines (static IPs).   
   
 network --device eth0 --bootproto static --ip 10.10.10.237   
 --netmask 255.255.255.0 --gateway 10.10.10.254   
 --nameserver 10.10.10.11,10.0.0.23,10.1.0.34   
 --hostname fc1-test   
   
 This line configures the interface eth0 with a static IP address of 10.10.10.237.   
 Notice that the nameserver selection accepts a comma-separated list of IP   
 addresses.   
   
 Configure other interfaces by specifying different devices with --device. You needn't  
 supply any network information when --bootproto is dhcp or bootp.   
 The network configuration will differ for each host in a clone farm, so you can't use  
 the same file for the entire group. I'll present ideas on how to handle this   
 situation in a future article.   
   
 Authentication and Security   
   
 Set the root password with the rootpw directive.   
   
 rootpw --iscrypted $1$NaCl$X5jRlREy9DqNTCXjHp075/   
   
 The --iscrypted flag indicates an MD5-hashed password. You can find a password's   
 encrypted form any number of ways, such as copying an existing entry from   
 /etc/shadow or using OpenSSL's passwd module:   
   
 $ openssl passwd -1 -salt "NaCl" "don't use this"   
   
 Without the --iscrypted flag the specified password will be used as-is, such as:   
   
 rootpw plain\_text   
   
 On the subject of passwords, authconfig determines how to authenticate users. The   
 following line sets the target host to use MD5-hashed passwords from the local   
 /etc/passwd and /etc/shadow files:   
   
 authconfig --enableshadow --enablemd5   
   
 There are other authentication options, such as NIS, LDAP, or Kerberos 5.   
 The firewall directive sets up a rudimentary ruleset, useful for a machine that will   
 talk to the outside world:   
   
 firewall --enabled --trust=eth0 --http --ssh   
   
 Here, traffic from interface eth0 will be implicitly trusted. The firewall will   
 permit incoming SSH (port 22/tcp) and HTTP (80/tcp) traffic on all interfaces.   
   
 Specify firewall --disabled to manually configure the firewall later or to skip it   
 altogether.   
   
 Time Zone   
   
 Set the machine's time zone with the timezone directive:   
   
 timezone America/Chicago   
   
 Valid time zones are in the TZ column of the file /usr/share/zoneinfo/zone.tab.   
 Boot Loader   
   
 The bootloader directive sets the location of the GRUB boot loader. The sample ks.cfg  
 places it in the master boot record (MBR):   
   
 bootloader --location=mbr   
   
 If you don't want a boot loader, specify --location=none. Remove an old boot loader   
 from the MBR with the separate zerombr directive.  
   
 Disks   
   
 Disk setup is the most complex part of a ks.cfg because there are so many machine-   
 and environment-dependent choices. Note that the sample ks.cfg clears existing   
 partitions on the target machine, so be sure to backup any valuable data.   
   
 clearpart removes disk partitions.   
   
 clearpart --all --drives=sda --initlabel   
   
 clearpart can remove just Linux partitions (--linux) or all existing partitions   
 (--all). It removes partitions from all drives unless you've added the --drives   
 flag.   
  
 The --initlabel flag works for previously unused disks or disks with foreign   
 partition schemes: it clears out the old partitions and sets up a scheme that Linux  
 can understand.   
   
 Omit clearpart to preserve existing partition boundaries.   
   
 part sets up partitions. The sample ks.cfg uses a simple two-partition layout and has  
 a separate swap partition:   
   
 part /boot --fstype ext3 --size=100 --ondisk=sda --asprimary   
 part / --fstype ext3 --size=1024 --grow --ondisk=sda --asprimary   
 part swap --size=128 --grow --size=256 --ondisk=sda --asprimary   
   
 The first parameter specifies the mount point, here /boot, /, and swap. (Linux   
 doesn't really mount swap space, but that's a minor technicality.) Set the   
 file-system type with the --fstype flag. The sample uses ext3. Other options   
 include ext2 and vfat (aka Windows). Swap doesn't use a file-system type.   
 Specify a partition's size in megabytes using the --size flag. Specify the   
 partition's physical disk with the optional --ondisk flag. Mark your primary   
 partitions with --asprimary.   
   
 part's --onpart and --noformat flags preserve existing partitions between Kickstart   
 installs. For example, the following would mount the preexisting hda7 as /home:   
   
 part /home --fstype ext3 --size 1024 --onpart hda7 --noformat   
   
 Note that this won't shuffle data to another part of the disk if other partition   
 sizes change; it simply tells Kickstart to leave hda7's partition boundaries intact  
 and to skip creating a new file system there using mkfs.   
   
 The following is a simple one-disk LVM setup:   
   
 part /boot --fstype ext3 --size=75 --asprimary   
 part pv.00 --size=1 --grow --asprimary   
   
 volgroup vgroot pv.00   
 logvol / --name=root.fs --vgname=vgroot --size=1024   
 logvol swap --name=swap.vol --vgname=vgroot --size=256   
   
 The second part directive sets up a partition as an LVM physical volume (PV). The   
 --grow flag grows this partition to the maximum allowable size, so that you needn't  
 know the disk's size ahead of time. part still requires a size, though, so it uses   
 a bogus PV partition size of 1MB.   
   
 logvol is LVM's part equivalent: it accepts the logical volume's mount point and   
 size, in addition to the volume group to which it belongs. logvol's --name flag   
 names the volume.   
   
 Note that the generated /root/anaconda-ks.cfg on the target host has a commented-out   
 disk layout.   
 Rebooting   
   
 The reboot directive forces the target host to reboot when the installation   
 completes. Don't forget to remove the installation media, lest the machine reboot   
 right back into the installer.   
   
 Package Selection   
   
 The %packages directive specifies which RPMs to install on the target host. You may   
 select packages individually or en masse as groups. To specify a group, prefix the   
 name with the @ symbol and a space. Precede a name with a minus symbol (-) to   
 exclude that package from the group.   
   
 %packages   
 @ dialup   
 kernel   
 grub   
 e2fsprogs   
   
 The Fedora/base/comps.xml file, from the install media, defines package groups. I'll   
 describe this file in greater detail in a future article.   
 Kickstart installs packages in addition to those you select in order to resolve   
 dependencies. Use %packages's --ignoredeps flag to ignore package dependencies.   
 Package selection is another area in which it is easiest to perform a manual   
 installation once, then mine the resultant /root/anaconda-ks.cfg file for   
 information.  
  
 Build the Target Machine: Run the Kickstart   
   
 The hard work is done. Now boot the target machine from the Fedora media. At the   
 boot: prompt, enter:   
   
 linux ks=http://build-server/kickstart/ks.cfg   
   
 You will receive an error if the boot media does not match the version of Fedora   
 you're trying to install.   
   
 Unless you have DHCP available on the target machine's network, the installation will  
 pause for you to enter its IP configuration. This is fine for small deployments and  
 experiments, but a full, hands-off Kickstart infrastructure calls for DHCP or   
 bootp.   
   
 The installation will also pause for input if any required directives are missing   
 from ks.cfg.   
   
 Troubleshooting a Failed Install   
   
 The installer's error reporting can be cryptic. Messages refer to lines in Anaconda's  
 underlying Python scripts, not your ks.cfg.   
   
 Include the interactive directive to step through the installation using values from   
 ks.cfg as the defaults. You cannot test the root password this way, though; you   
 must enter that manually.  
   
 Going Beyond   
   
 My Kickstart R&D has certainly paid off: I no longer have to click through the full   
 Fedora installer and I can grab a tea while Kickstart does the hard work.   
 Hopefully, this article will help jump-start your own Kickstart projects.   
 There is a lot more to Kickstart than what I have described here. It supports several  
 customization points, including home-grown RPMs and pre-/post-install scripts. I'll  
 cover these and more in a future article.  
   
 Resources   
   
 # Sample ks.cfg   
   
 ##   
 ## sample config file for testing Kickstart   
 ## for OnLAMP.com article   
 ##   
 ## NOTE: Likely, you'll have to change several values   
 ## here to match your hardware.   
 ##   
 ##   
   
 ## - - install type/source - - - - - - - - - - -   
 install   
 url --url http://build-server/FC1-install   
   
 ## - - debugging - - - - - - - - - -   
 ## :: uncomment the following to debug a Kickstart config file   
 ## interactive   
   
 ## - - language and input support - - - - - - - -   
 ## :: language used during install   
 lang en\_US.UTF-8   
   
 ## :: mouse used during install   
 mouse generic3ps/2   
   
 ## :: runtime language and keyboard support   
 langsupport --default en\_US.UTF-8 en\_US.UTF-8   
 keyboard us   
   
 ## - - video card and monitor - - - - - - - - - -   
   
 xconfig --card "VMWare" --videoram 16384 --hsync 31.5-37.9 --vsync 50-70 --resolution  
 800x600 --depth 16   
   
   
 ## - - network configuration - - - - - - - - - -   
 network --device eth0 --bootproto static --ip 10.10.10.237 --netmask 255.255.255.0   
 --gateway 10.10.10.254 --nameserver 10.10.10.11,10.0.0.23,10.1.0.34 --hostname   
 fc1-test   
   
   
 ## - - security and authentication - - - - - - -   
   
 rootpw --iscrypted $1$NaCl$X5jRlREy9DqNTCXjHp075/   
   
 firewall --disabled   
   
 authconfig --enableshadow --enablemd5   
   
 ## - - time zone - - - - - - - - - -   
 timezone America/Chicago   
   
 ## - - boot loader- - - - - - - - - -   
 bootloader --location=mbr   
   
 ## - - disk setup - - - - - - - - - -   
   
 ## :: remove old partitions   
   
 clearpart --all --initlabel   
   
 ## :: choose between hard partitioning and LVM,   
 ## :: then uncomment the proper set of lines   
   
 ## :: hard partitioning   
 # part /boot --fstype ext3 --size=100 --ondisk=sda   
 # part / --fstype ext3 --size=1024 --grow --ondisk=sda   
 # part swap --size=128 --grow --size=256 --ondisk=sda   
   
 ## :: LVM   
 # part /boot --fstype ext3 --size=100 --asprimary --ondisk=sda   
 # part pv.00 --size=1 --grow --asprimary --ondisk=sda   
 #   
 # volgroup vgroot pv.00   
 # logvol / --name=root.fs --vgname=vgroot --size=1024   
 # logvol swap --name=swap.vol --vgname=vgroot --size=256   
   
   
 ## - - package selection - - - - - - - - - -   
 ## :: reboot the machine when done   
 ## :: (it's up to you to remove the boot media)   
   
 reboot   
   
 ## - - package selection - - - - - - - - - -   
 ## :: this is a barebones install, just for testing Kickstart   
 %packages   
 @ dialup   
 kernel   
 grub   
 e2fsprogs   
   
 ## - - - - - - - - - - - - - - - - - - - - -

======================================   
 FILE SYSTEM ////////////////////////   
 ======================================   
   
   
   
   
 ======================================   
 Super Block / Data Block / Index:   
 ======================================   
   
 The [super block] contains an array of free disk block numbers, one of which points   
 to the next entry in the list. That entry in turn will be a [data block], which   
 contains an array of some other free blocks and a next pointer. When a process   
 requests a block, it searches the free block list, returns the available disk block  
 from the array of free blocks in the super block.   
   
 If the [super block] contains only one entry which is a pointer to a [data block],   
 which contains a list of other free blocks, all the entries from that block will be  
 copied to the [super block] free list and returns that block to the process.   
 Freeing of a block is reverse process of allocation. If the list of free blocks in   
 [super block] has enough space for the entry then, this block address will be   
 marked in the list.   
   
 If the list is full, all the entries from the [super block] will be copied to the   
 freed block and mark an entry for this block in the super block. Now the list in   
 super block contains only this entry. Index indexes to the next free disk block in   
 the free disk block list.   
   
 fsck -F ufs -o b=97472 /dev/rdsk/c0t0d0s0   
 /\* Check and repair a UFS filesystem on c0t0d0s0, using an alternate superblock \*/   
   
 newfs -Nv /dev/rdsk/c0t0d0s1   
 /\* To view the super blocks available \*/   
   
 [ root@enterprise ]$ newfs -Nv /dev/rdsk/c0t0d0s1   
 mkfs -F ufs -o N /dev/rdsk/c0t0d0s1 1049760 135 16 8192 1024 16 10 120 2048 t 0 0 8   
 128 n   
 /dev/rdsk/c0t0d0s1: 1049760 sectors in 486 cylinders of 16 tracks, 135 sectors   
 512.6MB in 31 cyl groups (16 c/g, 16.88MB/g, 8128 i/g)   
 super-block backups (for fsck -F ufs -o b=#) at:   
 32, 34736, 69440, 104144, 138848, 173552, 208256, 242960, 277664, 312368,   
 726512, 761216, 795920, 830624, 865328, 900032, 934736, 969440, 1004144,   
 1038848,   
 [ root@enterprise ]$   
   
   
   
   
   
   
 ======================================   
 SETUID/SETGID:   
 ======================================   
 The setuid permission may be set by prefixing a permission set with the number four   
 (4) as shown in the following example:   
   
 # chmod 4755 suidexample.sh   
   
 The permissions on the suidexample.sh file should now look like the following:   
   
 -rwsr-xr-x 1 trhodes trhodes 63 Aug 29 06:36 suidexample.sh   
   
 It should be noticeable from this example that an s is now part of the permission set  
 designated for the file owner, replacing the executable bit. This allows utilities   
 which need elevated permissions, such as passwd.   
   
 To view this in real time, open two terminals. On one, start the passwd process as a   
 normal user. While it waits for a new password, check the process table and look at  
 the user information of the passwd command.   
   
 In terminal A:   
   
 Changing local password for trhodes   
 Old Password:   
   
 In terminal B:   
   
 # ps aux | grep passwd   
 trhodes 5232 0.0 0.2 3420 1608 0 R+ 2:10AM 0:00.00 grep passwd   
 root 5211 0.0 0.2 3620 1724 2 I+ 2:09AM 0:00.01 passwd   
   
 As stated above, the passwd is run by a normal user, but is using the effective UID   
 of root.   
   
 The setgid permission performs the same function as the setuid permission; except   
 that it alters the group settings. When an application or utility is ran with this   
 setting, it will be granted the permissions based on the group that owns the file,   
 not the user who started the process.   
   
 To set the setgid permission on a file, provide the chmod command with a leading two   
 (2) as in the following example:   
   
 # chmod 2755 sgidexample.sh   
   
 The new setting may be viewed as before, notice the s is now in the field designated   
 for the group permission settings:   
   
 -rwxr-sr-x 1 trhodes trhodes 44 Aug 31 01:49 sgidexample.sh   
   
 truss -f -p <pid of a shell>   
   
 /\* Using multiple windows, this can be used to trace setuid/setgid programs \*/   
  
  
   
 ======================================   
 HARD LINK/SOFT LINK:   
 ======================================   
 The difference between a hard link and a soft link~   
   
 Hard links: a hard link is a pointer to the file's i-node. For example, suppose that   
 we have a file a-file.txt that contains the string "The file a-file.txt":   
   
 % cat a-file.txt   
 The file a-file.txt   
 %   
   
 Now we use the ln command to create a link to a-file.txt called b-file.txt:   
   
 % ls   
 ./ ../ a-file.txt   
 % ln a-file.txt b-file.txt   
 % ls   
 ./ ../ a-file.txt b-file.txt   
   
 The two names a-file.txt and b-file.txt now refer to the same data:   
   
 % cat b-file.txt   
 The file a-file.txt   
 %   
   
 If we modify the contents of file b-file.txt, then we also modify the contents of   
 file a-file.txt:   
   
 % vi b-file.txt   
 ...   
 % cat b-file.txt   
 The file a-file.txt has been modified.   
 % cat a-file.txt   
 The file a-file.txt has been modified.   
 %   
   
 and vice versa:   
 % vi a-file.txt   
 ...   
 % cat a-file.txt   
 The file a-file.txt has been modified again!   
 % cat b-file.txt   
 The file a-file.txt has been modified again!   
 %   
   
 Soft links (symbolic links): a soft link, also called symbolic link, is a file that   
 contains the name of another file. We can then access the contents of the other   
 file through that name. That is, a symbolic link is like a pointer to the file's   
 contents. For instance, supposed that in the previous example, we had used the -s   
 option of the ln to create a soft link:   
   
 % ln -s a-file.txt b-file.txt   
   
 But what are the differences between the two types of links, in practice? Let us look  
 at an example that highlights these differences. The directory currently looks like  
 this (let us assume that a-file.txt b-file.txt are both hard links to the same   
 file):   
   
 % ls   
 ./ ../ a-file.txt b-file.txt   
   
 Let us first add another symbolic link using the -s option:   
   
 % ln -s a-file.txt Symbolicb-file.txt   
 % ls -F   
 ./ ../ a-file.txt b-file.txt Symbolicb-file.txt@   
   
 A symbolic link, that ls -F displays with a @ symbol, has been added to the   
 directory. Let us examine the contents of the file:   
   
 % cat Symbolicb-file.txt   
 The file a-file.txt has been modified again!   
   
 If we change the file Symbolicb-file.txt, then the file a-file.txt is also modified.   
 % vi Symbolicb-file.txt   
 ...   
 % cat Symbolicb-file.txt   
 The file a-file.txt has been modified a third time!   
 % cat a-file.txt   
 The file a-file.txt has been modified a third time!   
 % cat b-file.txt   
 The file a-file.txt has been modified a third time!   
 %   
   
 If we remove the file a-file.txt, we can no longer access the data through the   
 symbolic link Symbolicb-file.txt:   
   
 % ls -F   
 ./ ../ a-file.txt b-file.txt Symbolicb-file.txt@   
 % rm a-file.txt   
 rm: remove `a-file.txt'? y   
 % ls -F ./ ../ b-file.txt Symbolicb-file.txt@   
 % cat Symbolicb-file.txt   
 cat: Symbolicb-file.txt: No such file or directory   
   
 The link Symbolicb-file.txt contains the name a-file.txt, and there no longer is a   
 file with that name. On the other hand, b-file.txt has its own pointer to the   
 contents of the file we called a-file.txt, and hence we can still use it to access   
 the data.   
  
 % cat b-file.txt   
 The file a-file.txt has been modified a third time!   
   
 Although it may seem like symbolic links are not particularly useful, hard links have  
 their drawbacks. The most significant drawback is that hard links cannot be created  
 to link a file from one file system to another file on another file system. A Unix   
 file structure hierarchy can consist of several different file systems (possibly on  
 several physical disks). Each file system maintains its own information regarding   
 the internal structure of the system and the individual files on the system. Hard   
 links only know this system-specific information, which make hard links unable to   
 span file systems. Soft links, on the other hand, know the name of the file, which   
 is more general, and are able to span file systems.   
   
 For a concrete analogy, suppose that our friend Joel User is a student at both UBC   
 and SFU. Both universities assign him a student number. If he tries to use his UBC   
 student number at SFU, he will not meet with any success. He will also fail if he   
 tries to use his SFU student number at UBC. But if he uses his legal name, Joel   
 User, he will probably be successful. The student numbers are system-specific (like  
 hard links), while his legal name spans both of the systems (like soft links).   
 Here is an example that demonstrates a situation where a hard link cannot be used and  
 a symbolic link is needed. Suppose that we try to create a hard link from the   
 current working directory to the C header stdio.h.   
   
 % ln /usr/include/stdio.h stdio.h   
 ln: creating hard link `stdio.h' to `/usr/include/stdio.h': Invalid cross-device link   
 %   
   
 The ln command fails because stdio.h is stored on a different file system. If we want  
 to create a link to it, we will have to use a symbolic link:   
   
 % ln -s /usr/include/stdio.h stdio.h   
 % ls -l   
 lrwxrwxrwx 1 a1a1 guest 20 Apr 20 11:58 stdio.h -> /usr/include/stdio.h   
 % ls ./ ../ stdio.h@   
 %   
   
 Now we can view the file stdio.h just as if it was located in the working directory.   
 For example:   
   
 % cat stdio.h   
 /\* Copyright (c) 1988 AT&T \*/   
 /\* All Rights Reserved \*/   
   
 /\* THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T \*/   
 /\* The copyright notice above does not evidence any \*/   
 /\* actual or intended publication of such source code. \*/   
   
 /\*   
 \* User-visible pieces of the ANSI C standard I/O package.   
 \*/   
   
 #ifndef \_STDIO\_H   
 #define \_STDIO\_H   
 ...   
 %   
   
 The entire output of the cat command was not included to save space.   
 Note that the long listing (ls -l) of a soft link does not accurately reflect its   
 associated permissions. To view the permissions of the file or directory that the   
 symbolic link references, the -L options of the ls command can be used. For   
 example:   
   
 % ln -s /usr/include/stdio.h stdio.h   
   
 % ls -l stdio.h   
 lrwxrwxrwx 1 a1a1 undergrad 20 May 10 15:13 stdio.h -> /usr/include/stdio.h   
   
 % ls -l /usr/include/stdio.h   
 -rw-r--r-- 1 root bin 11066 Jan 5 2000 /usr/include/stdio.h   
   
 % ls -lL stdio.h   
 -rw-r--r-- 1 root bin 11066 Jan 5 2000 stdio.h   
   
   
  
 ======================================   
 File Manipulation Solaris:   
 ======================================   
   
 File Manipulation   
   
 dos2unix | -ascii <filename>   
   
 /\* Converts DOS file formats to Unix \*/   
 =======================================   
   
 fold -w 180   
   
 /\* To break lines to have maximum char \*/   
 =======================================   
   
 split [-linecount] [file]   
   
 /\* Split files into pieces \*/   
 =======================================   
   
 [vi] : %s/existing/new/g   
   
 /\* Search and Replace text in vi \*/   
 =======================================   
   
 [vi] :set list   
   
 /\* Show non-printing characters in vi \*/   
 =======================================   
   
 [vi] :set nu   
   
 /\* Set line numbers in vi \*/   
 =======================================   
   
 [vi] :set ts=[num]   
   
 /\* Set tab stops in vi \*/   
 =======================================   
   
   
   
   
 ======================================   
 File System Solaris:   
 ======================================   
   
 /sbin/uadmin x x   
   
 /\* Syncs File Systems and Reboots systems fast \*/   
 =======================================   
   
 awk ' END {print NR}' file\_name   
   
 /\* Display the Number of lines in a file \*/   
 =======================================   
   
 cat /dev/null > filename   
   
 /\* Zero's out the file without breaking pipe \*/   
 =======================================   
   
 cksum [filename]   
   
 /\* View the checksum value for the given file \*/   
 =======================================   
   
 dd if=/dev/rdsk/... of=/dev/rdsk/... bs=4096   
   
 /\* Make a mirror image of your boot disk \*/   
 =======================================   
   
 df -k | grep dg| awk '{print $6}' |xargs -n 1 umount   
   
 /\* Unmount all file systems in disk group dg \*/   
 =======================================   
   
 fsck -F ufs -o b=97472 /dev/rdsk/c0t0d0s0   
   
 /\* Check and repair a UFS filesystem on c0t0d0s0, using an alternate superblock \*/   
 =======================================   
   
 fsck -F ufs -y /dev/rdsk/c0t0d0s0   
   
 /\* Check a UFS filesystem on c0t0d0s0, repair any problems without prompting. \*/   
 =======================================   
   
 fsck -F ufs /dev/rdsk/c0t0d0s0   
   
 /\* Check a UFS filesystem on c0t0d0s0 \*/   
 =======================================   
   
 gzip -d -c tarball.tgz | (cd /[dir];tar xf - ) &   
   
 /\* Unpacking tarballs to diff location \*/   
 =======================================   
   
 gzip -dc file1.tar.gz | tar xf -   
   
 /\* Unpack .tar.gz files in place \*/   
 =======================================   
   
 ln [-fhns] <source file> <destination file>   
   
 /\* Creating hard links and soft links \*/   
 =======================================   
   
 ls -al | awk '$3 == "oracle" || $3 == "root" {print $9}'   
   
 /\* List all file names by testing owner \*/   
 =======================================   
   
 ls -l | sort +4n   
   
 /\* List files by size \*/   
 =======================================   
   
 ls -la | awk '{ print $5," ",$9 }' | sort -rn   
   
 /\* File sizes of current directory \*/   
 =======================================   
   
 ls -lR | awk '{total +=$5};END {print "Total size: " total/1024/1024 "MB" }'   
   
 /\* Recursive directory size calculations in MB \*/   
 =======================================   
   
 mkisofs -l -L -r -o [image-name].iso [directory]   
   
 /\* Create an ISO image of a directory \*/   
 =======================================   
   
 mount -F ufs -o rw,remount /   
   
 /\* Used to remount root to make it writeable \*/   
 =======================================   
   
 mount -o remount,logging /spare   
   
 /\* Re-mount the ro file system rw and turn on ufs logging \*/   
 =======================================   
   
 mount DOS fdisk partition from Solaris   
   
 /\* mount -f pcfs /dev/dsk/c0d0p1 /export/dos \*/   
 =======================================   
   
 mv [filename]{,.new\_suffix}   
   
 /\* Renaming file \*/   
 =======================================   
   
 pax -rw . /newdir   
   
 /\* Efficient alternative for copying directories \*/   
 =======================================   
   
 prtvtoc /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c0t1d0s2   
   
 /\* Cloning Partitiontables \*/   
 =======================================   
   
 rpm -q --queryformat '%{INSTALLPREFIX}\n' [packagename]   
   
 /\* [Linux] Locate binaries \*/   
 =======================================   
   
 tar cf - . | (cd /newdir ; tar xf -)   
   
 /\* Recursively copy files and their permissions \*/   
 =======================================   
   
 tar cvf filename.tar   
   
 /\* Create a tape (tar) archive \*/   
 =======================================   
   
 tar xvf filename.tar   
   
 /\* Extract a tape (tar) archive \*/   
 =======================================   
   
 X=$(wc -l < filename); echo $X   
   
 /\* Count number of lines in a file into a variable (ksh) \*/   
 =======================================   
   
 zcat <patch\_file.tar.Z | tar xvf -   
   
 /\* Extract the patch\_file that is a compressed tar file \*/   
 =======================================   
   
 zcat [cpio file] | cpio -itmv   
   
 /\* Show the contents of a compressed cpio \*/   
 =======================================   
   
 File Transfer   
   
 find . -depth | cpio -pdmv /path/tobe/copied/to   
   
 /\* Fast alternative to cp -pr \*/   
 =======================================   
   
 find . -follow | cpio -pdumL /path/tobe/copied/to   
   
 /\* Copy with symbolic links to be followed \*/   
 =======================================   
   
 get filename.suffix |"tar xf -"   
   
 /\* Undocumented Feature of FTP \*/   
 =======================================   
   
 Move any file(s) without actually touching them   
   
 /\* ssh cd /some/directory \&\& tar cf - | ssh cd /some/direstory \&\& tar xvf - \*/   
 =======================================   
   
 put "| tar cf - ." filename.tar   
   
 /\* Undocumented Feature of FTP \*/   
 =======================================   
   
 sendport   
   
 /\* FTP command for transferring large numbers of files within the same control   
 session \*/   
 =======================================   
   
 General   
   
 /bin/printf '%d\n' '0x<hex>'   
   
 /\* Converts hexadecimal number to decimal. \*/   
 =======================================   
   
 /usr/bin/catman -w   
   
 /\* Create windex databases for man page directories \*/   
 =======================================   
   
 echo 'obase=16;255' | bc   
   
 /\* Simple way to convert decimal to hex \*/   
 =======================================   
   
 FQ\_FILENAME=<fully\_qualified\_file\_name>; echo ${FQ\_FILENAME%/\*}   
   
 /\* Extract directory from fully-qualified file name. \*/   
 =======================================   
   
 mailx -H -u <username>   
   
 /\* List out mail headers for specified user \*/   
 =======================================   
   
 ps -ef | grep -i $@   
   
 /\* Access common commands quicker \*/   
 =======================================   
   
 set filec   
   
 /\* Set file-completion for csh \*/   
 =======================================   
   
 uuencode [filename] [filename] | mailx -s "Subject" [user to mail]   
   
 /\* Send files as attachments \*/   
 =======================================   
   
 xauth -f /home/${LOGNAME} extract - ${DISPLAY} | xauth merge -   
   
 /\* Allow root to xdisplay after su \*/   
 =======================================   
   
   
   
   
   
   
 ======================================   
 VIRTUALIZATION //////////////////////   
 ======================================   
   
   
   
   
 ======================================   
 Solaris Zones:   
 ======================================   
   
 3.3 Sample Zone Configuration and Bring-Up   
   
 Here is a quick sample zone configuration where the zone name is my-zone and the IPv4  
 address is 10.0.0.1:   
   
 global# zonecfg -z my-zone   
 zonecfg:my-zone> create   
 /\* default is sparse root model, See section 3.4 for details\*/   
 zonecfg:my-zone> set zonepath=/export/home/my-zone   
 zonecfg:my-zone> add net   
 zonecfg:my-zone:net> set address=10.0.0.1   
 zonecfg:my-zone:net> set physical=eri0   
 zonecfg:my-zone:net> end   
 zonecfg:my-zone> verify   
 zonecfg:my-zone> commit   
 zonecfg:my-zone> ^D   
   
 At this point, a zone configuration file, /etc/zones/my-zone.xml, has been created   
 containing the above parameters and several inherited-pkg-dir fields for   
 loopback-mounted file systems. Once a zone configuration file is established, the   
 global zone administrator uses zoneadm(1M) to install the zone configuration:   
   
   
 global# zoneadm -z my-zone install   
   
 At the completion of the zoneadm(1M) install command, a boot environment is created   
 with the live\_upgrade(5) facilities. Zone boot is similar to booting a regular   
 Solaris environment, except that zoneadm(1M) is used to create the zone runtime:   
   
 global# zoneadm -z my-zone boot   
   
 This boots the zone. The appropriate file systems are mounted inside the zone,   
 zoneadmd(1M) is started, and so on. When a zone is booted for the first time after   
 installation, it has no internal configuration for naming schemes, no locale or   
 time zone, no root password, and so on. It is necessary to access the zone's   
 console to answer the prompts and set these up. This should be done using the   
 zlogin(1M) command:   
   
 # zlogin -C my-zone   
   
 [connected to zone my-zone console]   
   
 3.4 Zone Root File System   
   
 Two ways exist to configure a non-global zone's root file system: whole-root model   
 and sparse-root model.   
   
 The whole-root model provides the maximum configurability by installing all of the   
 required and any selected optional Solaris software packages into the private file   
 systems of the zone. The advantages of this model include the ability for zone   
 administrators to customize their zone's file-system layout (for example, creating   
 a /usr/local) and add arbitrary unbundled or third-party packages. The   
 disadvantages of this model include the loss of sharing of text segments from   
 executables and shared libraries by the virtual memory system, and a much heavier   
 disk footprint -- approximately an additional 2 Gbyte -- for each non-global zone   
 configured as such. The global zone administrator uses the sub-command create -b of  
 zonecfg(1M) to create a zone with the whole root mode (or alternatively to remove   
 the inherited-pkg-dir directories in my-zone.xml).   
   
 The sparse-root model optimizes the sharing of objects by installing only a subset of  
 the root packages (those with the pkginfo(4) parameter SUNW\_PKGTYPE set to root)   
 and using read-only loopback file systems to gain access to other files. This is   
 similar to the way a diskless client is configured, where /usr and other file   
 systems are mounted over the network with NFS. By default with this model, the   
 directories /lib, /platform, /sbin and /usr are mounted as loopback file systems.   
 The advantages of this model are greater performance due to the efficient sharing   
 of executables and shared libraries, and a much smaller disk footprint for the zone  
 itself. The sparse-root model only requires approximately 100 Mbyte of file system   
 space for the zone itself.

======================================  
 NAME SERVICES/////////////////////  
======================================  
======================================  
 DNS:  
======================================  
======================================  
 DNS Setup and Configuration:  
======================================   
  
Domain name services resolves host names to the IP addresses of clients and vice   
versa. The domain name system provides a convenient way of finding computer systems  
on the network based on their name and IP address. With increased internet usage   
and globalization of companies setting up of dns servers has become a major   
responsibility of system administarators worldwide.   
  
The following article describes in simple steps how to setup a dns server. Though the  
article focuses on Solaris, any operating system which uses BIND can use the same   
procedures.   
  
1.0 Introduction to DNS  
2.0 Requirements for setting up dns server  
3.0 DNS server Installation  
4.0 named.conf file in DNS server configuration  
4.1 options statement in DNS server configuration  
4.2 Zone in DNS server configuration  
4.3 Logging in DNS server  
5.0 Zone File in DNS server  
6.0 DNS Client configuration  
7.0 Signals in DNS server process - named  
8.0 Next steps   
  
1.0 Introduction   
  
A domain name system is a hierarchical system where there is a top level domain   
serving sub domains and clients with names & IP address.   
  
The system that runs the name services to resolves names into IP addresses is called   
name server and the sofware is generally BIND (Berkley Internet Domain).   
  
The core process of DNS is a daemon called named. Depending on the role assigned, the  
name servers can be a primary, secondry, or caching only. The secondry server takes  
over when the primary is down and is updated automatically. The caching server   
provides only the caching information to the clients.   
  
Each domain or sub domain has information (in zone files or data files) about its   
clients and is called authorative for these clients. The other clients for which it  
doesn't have any information or it is not authorative, it passes query to its   
higher domain.   
  
The client knows about their name servers through a file called resolve.conf which   
contains addresses of the name servers (Primary, secondary, and Caching) along with  
their domain name.   
  
The main file of the server is named.conf which contains server parameters and   
reference to other data files containing client information.   
  
2.0 Requirements:   
  
1) BIND (Berkely Internet Domain) software. Source code can be downloaded and   
compiled for your platform from internet at www.isc.org However BIND may be   
available in a precompiled version along with your OS, so check your OS if it is   
already there. The situation where you may want to compile from source code is that  
you want to cutomize it differently by giving different configuration options at   
compiling time.   
  
2) Root cache file from internic at ftp://internic.com/pub/root   
  
3) C Compiler to compile the bind source distribution.   
  
3.0 Installation and configuration   
  
Download the BIND software from from www.isc.org if you want to build it from source   
code.  
Make a directory to store and compile DNS distribution source say /usr/dns/src  
Unzip the distribution using gzip command   
  
# tar -zxvf bind-9.2.5.tar.gz   
  
Compilation requires a C compiler, if you don't have one you can download from gnu   
site (www.gnu.org).   
  
# ./configure   
  
# make   
  
# make install   
  
"make install" will ultimately place named, configuration file named.conf, and   
related commands in /etc and /usr/local/bin directory.   
  
4.0 named.conf file  
This is the main configuration file in BIND which defines the name servers and zones   
with the name and ip address of the hosts.   
  
The named.conf has a number of options for starting the name server which can be   
configured as per requirement. A list of complete options can be seen using 'man   
named' command.   
  
By default you will find zone files for local host by the name localhost and   
127.0.0.in-addr.arpa. For additional zones you need to create the the files and put  
a reference in named.conf.   
  
Below is a basic functional named.conf file which is installed after BIND 8.2.P5 is   
installed. This can be used for starting name server, all you need to do is to put   
your hosts entries in the zone files referenced here. You will find explanation of   
terms used in this configuration file after this listing of named.conf.   
  
// This is a configuration file for named (from BIND 8.1 or later).  
// It would normally be installed as /etc/named.conf.   
  
options { directory "/var/named";  
check-names master warn; /\* default. \*/  
datasize 20M;  
deallocate-on-exit yes;  
listen-on {10.20.30.100;  
};  
forward first;  
};  
zone "localhost" IN {  
type master;  
file "/var/named/localhost.zone";  
check-names fail;  
allow-update { none; };  
allow-transfer { any; };  
};  
zone "0.0.127.in-addr.arpa" IN {  
type master;  
file "/var/named/127.0.0.zone";  
check-names fail;  
allow-update { none; };  
allow-transfer { any; };  
};  
zone "." IN {  
type hint;  
file "/var/named/root.hint";  
};  
logging {  
channel xfer-log {  
file "/var/tmp/bind-xfer.log" versions unlimited size 10m;  
print-category yes;  
print-severity yes;  
print-time yes;  
severity info;  
};  
category xfer-in { xfer-log; };  
category xfer-out { xfer-log; };  
category notify { xfer-log; }   
  
category load { xfer-log; };  
};  
zone "30.20.10.in-addr.arpa" IN {  
type master;  
file "/var/named/100.30.20.10.zone";  
check-names fail;  
allow-update { none; };  
allow-transfer { any; };  
};  
zone "mydomain.com" {  
type master;  
file "/var/named/mydomain.com.hosts";  
};   
  
Explanation of the terms used in named.conf above  
  
   
  
4.1 Options statement  
The options statement lists working directory for named (the name server daemon) to   
read the configurations files and port to listen on (default is port 53).   
  
{ directory "/var/named";   
  
This directive defines the working dir of the name server where main configuration   
file named.conf will be located   
  
check-names master warn; /\* default. \*/   
  
The "check-names" directive tells BIND to check names in master zone and gives a   
warning in the system's log files if there is any discrepancy. Names are considered  
good if they match RFC 952's expectations (if they are host names), or if they   
consist only of printable ASCII characters (if they are not host names).   
  
Other options are fail and ignore in that case bind will follow these directives   
  
datasize 20M;   
  
Datasize is the maximum amount of data memory the server may use. The default is   
system dependent.   
  
deallocate-on-exit yes;   
  
Deallocate the memory on exit, otherwise it will be left to the OS to clear the   
memory.   
  
listen-on {10.20.30.100};   
  
Host address and port for listening; if port is not mentioned it is default 53.   
  
forward first   
  
Forwarding  
Forwarding can be used for two main scenarios:  
1. Creating a large site wide cache on different servers thereby using less network   
bandwidth.   
  
2. For servers which do not have a direct access to the internet but have to lookup   
for the external names.   
  
Forwarding occurs only for names for which the server is not authoritative, and it   
does not have the answer in its cache.   
  
forward  
This option specify where to query the name first - 'first' directive will cause   
query to send to forwarder first and check itself if it fails. 'Only' - directive   
will query the forwarders only.   
  
forwarders  
Specifies the IP addresses to be used for forwarding. The default is no forwarding .  
  
   
  
4.2 Zones statements   
  
zone "localhost" IN {  
type master;  
file "/var/named/localhost.zone";  
check-names fail;  
allow-update { none; };  
allow-transfer { any; };  
};   
  
Zone statement declares a zone name, its type - master, slave, or stub files   
containing the zone data, and options relating to zone - update, checking, transfer  
etc.   
  
localhost and 0.0.127.in-addr.arpa are default for the localhost and points to file   
of this name   
  
Zone types   
  
There are three types of zones.   
  
master: This is the master copy of the data in a zone.   
  
slave: This is a replica of a master zone. The masters list specifies one or more IP   
addresses that the slave contacts to update its copy of the zone. If file is   
specified, then the replica will be written to the file. Use of file is   
recommended, since it often speeds server startup and eliminates a needless waste   
of bandwidth.   
  
stub: A stub zone is like a slave zone, except that it replicates only the NS records  
of a master zone instead of the entire zone.   
  
hint: The initial set of root name servers is specified using a hint zone. When the   
server starts up, it uses the root hints to find a root name server and get the   
most recent list of root name servers.   
  
The previous releases of BIND used the term primary for a master zone, secondary for   
a slave zone, and cache for a hint zone.   
  
Zone Directives   
  
allow-update  
Specifies which hosts are allowed to submit dynamic DNS updates to the server. The   
default is to deny updates from all hosts.   
  
allow-transfer  
Specifies which hosts are allowed to receive zone transfers from the server.   
allow-transfer may also be specified in the zone section, in which case it   
overrides the options allow-transfer statement. If not specified, the default is to  
allow transfers from all hosts.   
  
Zone "." refers to the root file for the domains - and contains references to the   
root servers at network solutions to resolve the names which are beyond the current  
domain. You can download the root cache file from ftp://internic.com/pub/root  
  
   
  
4.3 Logging statement   
  
logging {  
channel xfer-log {  
file "/var/tmp/bind-xfer.log" versions unlimited size 10m;  
print-category yes;  
print-severity yes;  
print-time yes;  
severity info;  
};   
  
The logging statement specifies logging channel/s which logs various categories of   
messages. In the statement above a channel xfer-log - a user defined name is   
defined. Each time name server is started it starts writing to the defined log   
file. Size limits the maximum size of the log file and once the limit is reached it  
stops writing the file. Each individual start or restart of named causes a new   
version of the log file to be created. Version statement defines how many versions   
are allowed for the log file. The unlimited option will allow any number of   
versions.   
  
Only one logging statement is used to define how many channels and categories are   
wanted. If there are multiple logging statements in a configuration, the first   
definition determines the logging and warnings are issued for the other logging   
statements.   
  
If there is no logging statement, the default logging configuration is used which   
is:   
  
logging {  
category default { default syslog; default\_debug;};  
category panic { default syslog; default\_stderr;};  
category packet { default\_debug;};  
category eventlib { default\_debug;};  
};   
  
The default debug file is named.run .   
  
Channel Phrase   
  
All log output goes to one or more "channels"; you can make as many of them as you   
want. Every channel definition must include a clause that says whether messages   
selected for the channel go to a file, to a particular syslog facility, or are   
discarded. It can optionally also limit the message severity level that will be   
accepted by the channel (default is "info"), and whether to include a   
named-generated time stamp, the category name and/or severity level (default is not  
to include any).   
  
The word null as the destination option for the channel will cause all messages sent   
to it to be discarded; other options for the channel are meaningless.   
  
The file clause defines size and versions of the file which will be saved each time   
the file is opened. if the file ever exceeds the size, then named will just not   
write anything more to it. The default behavior is to not limit the size of the   
file.   
  
As per selection the log messages will either go to syslog() or a file and severity   
level determines which type of messages goes there. Default severity level is info.  
and it can be critical, error, debug, and dynamic.   
  
Note that only syslog messages can go to syslog.   
  
Print-time, print-category - logs the time & category of the messages . The print-   
options can be used in any combination but will always be printed in the following   
order: time, category, severity.   
  
category xfer-in { xfer-log; };  
category xfer-out { xfer-log; };  
category notify { xfer-log; }   
  
These directives put diffrent categories of log messages in to xfer-log channel   
  
Category option mentions the category of the log and file name for logging   
  
logging {  
channel xfer-log {  
file "/var/tmp/bind-xfer.log" versions unlimited size 10m;  
print-category yes;  
print-severity yes;  
print-time yes;  
severity info;  
};   
  
this defines a channel called xfer-log with various options.   
  
these categories directs various types of logs into the channel  
  
   
  
5.0 ZONE files   
  
Zone files are used to define the name and ip addresses of the hosts in a domain.   
Generally two zone files are defined for a particular zone - one file maps the the   
name to the IP address of the host machines and the other is used for reverse   
lookup i.e.., IP address to name address.   
  
Each master zone file should begin with an SOA (Start of Authority) record for the   
zone. The SOA specifies a serial number, which should be changed each time the   
master file is changed. The serial number has a 32 bit size field. Slave servers   
check the serial number at refresh time and if they detect a changed serial number   
in the master, then the zone transfer is carried out to keep its zone files   
updated.   
  
If a master server cannot be contacted within the interval given by the expire time,   
all data from the zone is discarded by slave servers. The minimum value is the   
time-to-live (``TTL'') used by records in the file with no explicit time-to-live   
value.   
  
The details of all type of records used in a zone file are given below:   
  
Type of records:  
SOA marks the start of a zone of authority (domain of originating host, domain   
address of maintainer, a serial number and the following parameters in seconds:   
refresh, retry, expire and minimum TTL. (see RFC 883).   
  
NULL a null resource record (no format or data)   
  
RP a Responsible Person for some domain name   
  
PTR a domain name pointer (domain)   
  
HINFO host information (cpu\_type OS\_type)   
  
A a host address (dotted quad)   
  
NS an authoritative name server (domain)   
  
MX a mail exchanger (domain), preceded by a preference value (0..32767), with lower   
numeric values representing higher logical preferences.   
  
CNAME the canonical name for an alias (domain).   
  
Following are the three functional zone files representing local host and a master   
zone.   
  
The explanation of the terms are at the end.   
  
/var/named/localhost   
  
localhost. 1D IN SOA localhost.mydomainr.com. hostmaster.mydomain.com. (  
42 ; serial  
3H ; refresh  
15M ; retry  
1W ; expiry  
1D ) ; minimum  
localhost. NS dns  
localhost. A 127.0.0.1   
  
/var/named/ 0.0.127.in-addr.arpa   
  
0.0.127.in-addr.arpa IN SOA localhost. root.localhost. (  
42 ; serial  
3H ; refresh  
15M ; retry  
1W ; expiry  
1D ) ; minimum   
  
0.0.127.in-addr.arpa IN NS dns.mydomain.com  
1.0.0.127.in-addr.arpa. PTR localhost   
  
/var/named/mydomain.com   
  
mydomain.com. IN SOA dns.mydomain.com hostmaster.dns. (  
200010016 ;serial  
10800  
3600  
3600  
86400 )  
mydomain.com. 1D IN NS dns.mydomain.com.  
IN MX 20 mx1.domaingateway.net.  
IN MX 10 mail-in.mydomain.com.   
  
;mydomain hosts below  
www IN CNAME mydomain.com.  
localhost IN A 127.0.0.1  
mail IN A xxx.xxx.xxx.xxx  
ns1 IN A xxx.xxx.xxx.xxx  
dns IN A xxx.xxx.xxx.xxx  
news IN A xxx.xxx.xxx.xxx   
  
root cache file   
  
localhost. NS dns   
  
this is declaration of the type of localhost it declares that local host is a name   
server with hostname dns   
  
localhost. A 127.0.0.1   
  
this declares the address of local host.   
  
0.0.127.in-addr.arpa IN NS dns.mydomain.com  
1.0.0.127.in-addr.arpa. PTR localhost   
  
Similarly in reverse zone map file reverse address is declared as ns record of name   
dns and a pointer record (ptr), points this reverse address to the localhost.   
  
Resource records normally end at the end of a line, but may be continued across lines  
between opening and closing parentheses. Comments are introduced by semicolons and   
continue to the end of the line.Note that there are other resource record types,   
not shown where. You should consult the BIND Operations Guide (BOG') for the   
complete list. Some resource record types may have been standardized in newer RFC's  
but not yet implemented in this version of BIND.  
  
   
  
6.0 Client Configuration   
  
Each client need a configuration file /etc/resolv.conf which informs it about the   
domain name server. This is a editable text file with following entries:   
  
domainname yourdomainname.com  
nameserver 10.20.30.40  
nameserver 10.20.30.41   
  
7.0 Signals   
  
The following signals have the specified effect when sent to the server process named  
using the kill command.   
  
SIGHUP:  
Causes server to read named.boot and reload the database. If the server is built with  
the FORCED\_RELOAD compile-time option, then SIGHUP will also cause the server to   
check the serial number on all secondary zones. Normally the serial numbers are   
only checked at the SOA-specified intervals.   
  
SIGINT:  
Dumps the current data base and cache to /var/named/named\_dump.db   
  
SIGIOT:  
Dumps statistics data into /var/named/named.stats. If the server is compiled with   
-DSTATS. Statistics data is appended to the file. Some systems use SIGABRT rather   
than SIGIOT for this.   
  
SIGSYS:  
Dumps the profiling data in /var/named if the server is compiled with profiling   
(server forks, chdirs and exits).   
  
SIGTERM:  
Dumps the primary and secondary database files. Used to save modified data on   
shutdown if the server is compiled with dynamic updating enabled.   
  
SIGUSR1:  
Turns on debugging; each SIGUSR1 increments debug level. (SIGEMT on older systems   
without SIGUSR1)   
  
SIGUSR2:  
Turns off debugging completely. (SIGFPE on older systems without SIGUSR2)   
  
SIGWINCH:  
Toggles logging of all incoming queries via sys-log(8) (requires server to have been   
built with the QRYLOG option)  
  
  
======================================  
 Zone File:  
======================================   
  
If you make updates to a zone file but leave the serial number unchanged, what   
happens?~   
  
Serial number: Now this value has special purpose. For optimal speed and efficiency,   
BIND (one of the most widely used DNS servers, designed for Unix-based operating   
systems) processes zone files into a different format. What happens is, when BIND   
loads a zone file at startup, it first looks up at its serial number and proceeds   
with processing the zone file only if its serial number is bigger than the last   
processed version. Thus, if you make modifications to a zone file, but leave   
unchanged its serial number, BIND will ignore your updated version.  
The typical format of a serial number comprises date and unique serial number   
(YYYYMMDDNN), such as: 2009010801 - for the second edition (01) of the file on   
January 8, 2009. This format allows 100 modifications to be made to the zone file   
per day.  
  
  
======================================  
 NIS:  
======================================  
======================================  
 autofs / automounter / files / NIS Solaris:  
======================================   
  
3.1: How to Set Up a SunOS Automount Client Using Files   
  
Under SunOS, the automounter is centered around the file  
/etc/auto.master. This file must contain a number of lines in the  
following format:   
  
directory mapname options   
  
Where: directory is the directory to mount an indirect map in, or /-  
for a direct map mapname is the file which contains the map and  
options are any standard NFS options which should be used for the  
entire map. An example of an auto.master file follows:   
  
% cat /etc/auto.master  
/- /etc/auto.direct -ro  
/home /etc/auto.home   
  
In this example, /etc/auto.direct will be a direct map, which mounts a  
number of filesystems readonly, while /etc/auto.home will be a  
indirect map, which mounts filesystems under the /home directory.   
  
A typical map contains a number of lines as follows:   
  
mountpoint [options] remotemachine:/remotelocation   
  
The [options] can be omitted if only the standard options should be  
used. montpoint will be a full path for a direct mount (/usr/local) or  
just a directory name for an indirect mount (joe).   
  
For example, the auto.direct map may read:   
  
% cat /etc/auto.direct  
/usr/man -soft server:/usr/man  
/usr/local server:/export/sunos/usr/local   
  
This would create automount points for /usr/man and /usr/local, and  
/usr/man would be mounted soft.   
  
While the auto.home map (or any indirect map) would look something  
like this:   
  
% cat /etc/auto.home  
joe server:/export/home/joe  
fred server:/export/home/fred   
  
This would create automount points for /home/joe and /home/fred.   
  
Please note that automount will use an auto.master NIS map by default.  
Thus, to force automounter to use local files, you must start is as  
follows:   
  
# automount -f /etc/auto.master &   
  
You will also want to modify the automount startup in /etc/rc.local.   
  
3.2: How to Set Up a SunOS Automount Client Using NIS   
  
To force your automounter to read in NIS maps, you must change the way  
that the mapname is referenced in your auto.master file. If the  
mapname is listed without any "/"s, NIS maps will be automatically  
checked. The following auto.master file says to get the direct listing  
from the auto.direct NIS map, and the /home listing from the auto.home  
NIS map:   
  
# cat /etc/auto.master  
/- auto.direct -ro  
/home auto.home   
  
(Compare this to Section 3.1, where the mapname column of the  
/etc/auto.master map contains "/"s, directing the automounter to a  
local path.)   
  
In order to get a SunOS client to start automount, using the NIS maps,  
all you need to do is either create an auto.master map in NIS, and  
distribute it (See Section 3.5), or create a local map, as noted  
above. Other maps should be created on the NIS master, with the same  
format as is described in Section 3.1 (see Section 3.5 for how to  
modify those NIS maps).   
  
Afterwards, simply reboot the machine, or start up automount:   
  
# automount &   
  
[As a note, you may also read in NIS maps by putting a +mapname entry,  
ie +auto.home, in a local file this is usually done to set up a  
unique automounter on a certain machine. The references in Section 7.0  
should be used if you wish to implement a more complex set up, such as  
this.]   
  
3.3: How to Set Up a Solaris Automount Client Using Files   
  
Follow the instructions in Section 3.1, but be aware that under  
Solaris, the names of the files are expected to contain "\_"s instead  
of "."s. ie:   
  
/etc/auto\_master  
/etc/auto\_direct  
/etc/auto\_home   
  
So, using only files, your auto\_master should look something like  
this:   
  
# cat /etc/auto\_master  
/- /etc/auto\_direct -ro  
/home /etc/auto\_home   
  
The other files would follow with the same format as described in  
Section 3.1.   
  
When everything is set up, you can get automount starting by rebooting  
the machine, or running:   
  
# /etc/init.d/autofs start   
  
  
  
3.4: How to Set Up a Solaris Automount Client Using Other Naming Services   
  
If the /etc/auto\_master file contains mapnames without "/"s,  
additional naming services are consulted, according to the order  
listed in the nsswitch.conf. For example, the following  
/etc/nsswitch.conf line would say to check first files, then NIS:   
  
automount: files nis   
  
In addition, the local files may say to read other naming services, by  
listing the entry "+mapname".   
  
Following is an extremely typical automount setup for Solaris:   
  
# cat /etc/auto\_master  
+auto\_master  
/net -hosts -nosuid  
/home auto\_home   
  
[The +auto\_master line says to first check naming services (NIS/NIS+)  
for an auto\_master map. Afterwards, it includes a special net map,  
which is described in the man page, and also a /home indirect map,  
which is read from the naming services.]   
  
# cat /etc/auto\_home  
+auto\_home   
  
[This file says to just go out to naming services. It is necessary  
because "files" is one of the options listed in the nsswitch.conf.]   
  
# ypcat auto\_master  
[any additional auto\_master entries are listed here]  
# ypcat auto\_home  
...  
[the full auto\_home map is here]   
  
[Thus you will need to setup all of your normal maps in NIS or NIS+,  
as is described in Sections 3.5 and 3.6.]   
  
Of special note here is this: If NIS is listed as the naming service,  
and automountd can't find an auto\_map, then it will try instead to  
lookup auto.map, since that is the older NIS standard. So, the above  
would work fine if you were using NIS, and the actual NIS map was  
auto.home.   
  
When everything is set up, you can get automount started by rebooting  
the machine, or running:   
  
# /etc/init.d/autofs start  
  
   
  
3.5: How to Modify Automount Maps Under NIS   
  
The auto.master and auto.home maps are automatically part of NIS. To  
distribute these maps, simply edit the files /etc/auto.master and  
/etc/auto.home on the master, using the format described in Section  
3.1, and then make the maps to distribute them:   
  
# cd /var/yp  
# make   
  
Section 3.7 describes how to create new NIS maps.  
  
   
  
3.6: How to Modify Automount Maps Under NIS+   
  
The auto\_master and auto\_home tables are automatically part of NIS+.  
They may be modified, using the format described in Section 3.1. The  
auto\_home table may be modified via admintool, nistbladm or nisaddent  
(admintool is suggested). The auto\_master table may be modified via  
nistbladm or nisaddent. nisaddent is probably the best options for  
making this modification.   
  
To make a modification with nisaddent, you should first dump your map  
to a text file:   
  
# /usr/lib/nis/nisaddent -d -t auto\_master.org\_dir key-value >  
/etc/auto\_master.nisplus   
  
Then, you can edit the file with your favorite editor. Remember to use only  
tabs between the fields, not embedded spaces:   
  
# cat /etc/auto\_master.nisplus  
+auto\_master  
/net -hosts -nosuid  
/home auto\_home   
  
Afterwards, run nisaddent again to replace the NIS+ map with your text file:   
  
# /usr/lib/nis/nisaddent -r -f /etc/auto\_master.nis -t auto\_master.org\_dir  
key-value   
  
Section 3.8 describes how to create new NIS+ maps.   
  
  
3.7: How to Create New NIS Automount Maps   
  
Automount Maps   
  
The following example explains how to create an auto\_direct map under  
NIS. Other new maps can be created with similar syntax.   
  
In order to create an auto.direct map, you need to make a new entry in  
/var/yp/Makefile for auto.direct, mimicking the already existing  
auto.home entry:   
  
auto.direct.time: $(DIR)/auto.direct  
-@if [ -f $(DIR)/auto.direct ] then \  
sed -e "/^#/d" -e s/#.\*$$// $(DIR)/auto.direct \  
$(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.direct \  
touch auto.direct.time \  
echo "updated auto.direct" \  
if [ ! $(NOPUSH) ] then \  
$(YPPUSH) auto.direct \  
echo "pushed auto.direct" \  
else \  
: \  
fi \  
else \  
echo "couldn't find $(DIR)/auto.direct" \  
fi   
  
NOTE: all INDENTED $lines in the Makefile entry MUST be indented  
with the TAB key, without any imbedded spaces!!   
  
In addition, auto.direct must be added to the all: line, near the top  
of the Makefile:   
  
all: passwd group hosts ethers networks rpc services protocols \  
netgroup bootparams aliases publickey netid netmasks c2secure \  
timezone auto.master auto.home auto.direct   
  
And, finally, near the bottom, the following line must be added:   
  
auto.direct: auto.direct.time   
  
[Be very careful if you just copy the above lines -- Makefile entries  
MUST begin with TABS, not spaces if you text copy the above, you will  
end up with spaces at the beginning of each line, and make will fail.]   
  
When this is all done, you may create an /etc/auto.direct map, put the  
appropriate files in it, and then do a Make:   
  
# cd /var/yp  
# make   
  
After you have done the first make, you will probably gets some errors  
like the following:   
  
"can't bind master to send ypclear message to ypserv for map ..."   
  
This occurs because NIS is confused due to the maps not existing on  
the slave machines. To resolve this, you must manually copy the map to  
the slaves. This can be done by copying /var/yp/`domainname`/auto.direct.\*  
from the master to /var/yp/`domainname` on each of the slaves, using  
either rcp or ftp.   
  
Afterwards, do a second make:   
  
# cd /var/yp  
# make   
  
  
3.8: How to Create New NIS+ Automount Maps   
  
The following example explains how to create an auto\_direct map under  
NIS+. Other new maps can be created with similar syntax.   
  
STEP ONE: create a new auto\_direct table, on the master server:   
  
# nistbladm -c automount\_map key=S value=S auto\_local.org\_dir.`domainname`.   
  
STEP TWO: set the group ownership of the table:   
  
# nischgrp admin.`domainname`. auto\_local.org\_dir   
  
STEP THREE: set the correct permissions.   
  
# nischmod n=r, o=rmcd,g=rmcd, w=r auto\_local.org\_dir   
  
STEP FOUR: create a text file, and read it into NIS+, just as is  
described in section 3.6:   
  
# cat /etc/auto\_local.nisplus  
/usr/local/bin server:/usr/local/bin  
/usr/local/lib server2:/usr/local/lib  
# /usr/lib/nis/nisaddent -r -f /etc/auto\_local.nisplus \  
-t auto\_local.org\_dir key-value   
  
STEP FIVE: verify the data is in the map:  
# niscat -m auto\_local.org\_dir   
  
(your data should display)   
  
NOTE: You will also want to add an entry to your NIS+ auto\_master map, as is  
described in 3.6.  
  
  
======================================  
 Configure NIS Linux:  
======================================   
  
You need to add the NIS domain you wish to use in the /etc/sysconfig/network file.   
For the school, call the domain NIS-SCHOOL-NETWORK.   
  
#/etc/sysconfig/network  
NISDOMAIN="NIS-SCHOOL-NETWORK"   
  
NIS servers also have to be NIS clients themselves, so you'll have to edit the NIS   
client configuration file /etc/yp.conf to list the domain's NIS server as being the  
server itself or localhost.   
  
# /etc/yp.conf - ypbind configuration file  
ypserver 127.0.0.1   
  
Start the necessary NIS daemons in the /etc/init.d directory and use the chkconfig   
command to ensure they start after the next reboot.   
  
[root@bigboy tmp]# service portmap start  
Starting portmapper: [ OK ]  
[root@bigboy tmp]# service yppasswdd start  
Starting YP passwd service: [ OK ]  
[root@bigboy tmp]# service ypserv start  
Setting NIS domain name NIS-SCHOOL-NETWORK: [ OK ]  
Starting YP server services: [ OK ]  
[root@bigboy tmp]#   
  
[root@bigboy tmp]# chkconfig portmap on  
[root@bigboy tmp]# chkconfig yppasswdd on  
[root@bigboy tmp]# chkconfig ypserv on   
  
Table 30-1 Required NIS Server Daemons  
Daemon  
portmap The foundation RPC daemon upon which NIS runs.  
yppasswdd Lets users change their passwords on the NIS server from NIS clients  
ypserv Main NIS server daemon  
ypbind Main NIS client daemon  
ypxfrd Used to speed up the transfer of very large NIS maps   
  
Make sure they are all running before continuing to the next step. You can use the   
rpcinfo command to do this.   
  
[root@bigboy tmp]# rpcinfo -p localhost  
program vers proto port  
100000 2 tcp 111 portmapper  
100000 2 udp 111 portmapper  
100009 1 udp 681 yppasswdd  
100004 2 udp 698 ypserv  
100004 1 udp 698 ypserv  
100004 2 tcp 701 ypserv  
100004 1 tcp 701 ypserv  
[root@bigboy tmp]#   
  
Now that you have decided on the name of the NIS domain, you'll have to use the   
ypinit command to create the associated authentication files for the domain. You   
will be prompted for the name of the NIS server, which in this case is bigboy.   
  
With this procedure, all nonprivileged accounts are automatically accessible via   
NIS.   
  
[root@bigboy tmp]# /usr/lib/yp/ypinit -m  
At this point, we have to construct a list of the hosts which will run NIS  
servers. bigboy is in the list of NIS server hosts. Please continue to add  
the names for the other hosts, one per line. When you are done with the  
list, type a <control D>.  
next host to add: bigboy  
next host to add:  
The current list of NIS servers looks like this:   
  
bigboy   
  
Is this correct? [y/n: y] y  
We need a few minutes to build the databases...  
Building /var/yp/NIS-SCHOOL-NETWORK/ypservers...  
Running /var/yp/Makefile...  
gmake[1]: Entering directory `/var/yp/NIS-SCHOOL-NETWORK'  
Updating passwd.byname...  
Updating passwd.byuid...  
Updating group.byname...  
Updating group.bygid...  
Updating hosts.byname...  
Updating hosts.byaddr...  
Updating rpc.byname...  
Updating rpc.bynumber...  
Updating services.byname...  
Updating services.byservicename...  
Updating netid.byname...  
Updating protocols.bynumber...  
Updating protocols.byname...  
Updating mail.aliases...  
gmake[1]: Leaving directory `/var/yp/NIS-SCHOOL-NETWORK'   
  
bigboy has been set up as a NIS master server.   
  
Now you can run ypinit -s bigboy on all slave server.  
[root@bigboy tmp]#   
  
Note: Make sure portmap is running before trying this step or you'll get errors, such  
as:   
  
failed to send 'clear' to local ypserv: RPC: Port mapper failureUpdating   
group.bygid...   
  
You will have to delete the /var/yp/NIS-SCHOOL-NETWORK directory and restart portmap,  
yppasswd, and ypserv before you'll be able to do this again successfully.   
  
You can now start the ypbind and the ypxfrd daemons because the NIS domain files have  
been created.   
  
[root@bigboy tmp]# service ypbind start  
Binding to the NIS domain: [ OK ]  
Listening for an NIS domain server.  
[root@bigboy tmp]# service ypxfrd start  
Starting YP map server: [ OK ]  
[root@bigboy tmp]# chkconfig ypbind on  
[root@bigboy tmp]# chkconfig ypxfrd on   
  
All the NIS daemons use RPC port mapping and, therefore, are listed using the rpcinfo  
command when they are running correctly.   
  
[root@bigboy tmp]# rpcinfo -p localhost  
program vers proto port  
100000 2 tcp 111 portmapper  
100000 2 udp 111 portmapper  
100003 2 udp 2049 nfs  
100003 3 udp 2049 nfs  
100021 1 udp 1024 nlockmgr  
100021 3 udp 1024 nlockmgr  
100021 4 udp 1024 nlockmgr  
100004 2 udp 784 ypserv  
100004 1 udp 784 ypserv  
100004 2 tcp 787 ypserv  
100004 1 tcp 787 ypserv  
100009 1 udp 798 yppasswdd  
600100069 1 udp 850 fypxfrd  
600100069 1 tcp 852 fypxfrd  
100007 2 udp 924 ypbind  
100007 1 udp 924 ypbind  
100007 2 tcp 927 ypbind  
100007 1 tcp 927 ypbind  
[root@bigboy tmp]#   
  
New NIS users can be created by logging into the NIS server and creating the new user  
account. In this case, you'll create a user account called nisuser and give it a   
new password.   
  
Once this is complete, you then have to update the NIS domain's authentication files   
by executing the make command in the /var/yp directory.   
  
This procedure makes all NIS-enabled, nonprivileged accounts become automatically   
accessible via NIS, not just newly created ones. It also exports all the user's   
characteristics stored in the /etc/passwd and /etc/group files, such as the login   
shell, the user's group, and home directory.   
  
[root@bigboy tmp]# useradd -g users nisuser  
[root@bigboy tmp]# passwd nisuser  
Changing password for user nisuser.  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@bigboy tmp]# cd /var/yp  
[root@bigboy yp]# make  
gmake[1]: Entering directory `/var/yp/NIS-SCHOOL-NETWORK'  
Updating passwd.byname...  
Updating passwd.byuid...  
Updating netid.byname...  
gmake[1]: Leaving directory `/var/yp/NIS-SCHOOL-NETWORK'  
[root@bigboy yp]#   
  
You can check to see if the user's authentication information has been updated by   
using the ypmatch command, which should return the user's encrypted password   
string.   
  
[root@bigboy yp]# ypmatch nisuser passwd  
nisuser:$1$d6E2i79Q$wp3Eo0Qw9nFD/::504:100::/home/nisuser:/bin/bash  
[root@bigboy yp]   
  
You can also use the getent command, which has similar syntax. Unlike ypmatch, getent  
doesn't provide an encrypted password when run on an NIS server, it just provides   
the user's entry in the /etc/passwd file. On a NIS client, the results are   
identical with both showing the encrypted password.   
  
[root@bigboy yp]# getent passwd nisuser  
nisuser:x:504:100::/home/nisuser:/bin/bash  
[root@bigboy yp]#   
  
Now that the NIS server is configured, it's time to configure the NIS clients. There   
are a number of related configuration files that you need to edit to get it to   
work. Take a look at the procedure.   
  
The authconfig or the authconfig-tui program automatically configures your NIS files   
after prompting you for the IP address and domain of the NIS server.   
  
[root@smallfry tmp]# authconfig-tui   
  
Once finished, it should create an /etc/yp.conf file that defines, amongst other   
things, the IP address of the NIS server for a particular domain. It also edits the  
/etc/sysconfig/network file to define the NIS domain to which the NIS client   
belongs.   
  
# /etc/yp.conf - ypbind configuration file  
domain NIS-SCHOOL-NETWORK server 192.168.1.100   
  
#/etc/sysconfig/network  
NISDOMAIN=NIS-SCHOOL-NETWORK   
  
In addition, the authconfig program updates the /etc/nsswitch.conf file that lists   
the order in which certain data sources should be searched for name lookups, such   
as those in DNS, LDAP, and NIS. Here you can see where NIS entries were added for   
the important login files.   
  
#/etc/nsswitch.conf  
passwd: files nis  
shadow: files nis  
group: files nis   
  
Note: You can also locate a sample NIS nsswitch.conf file in the   
/usr/share/doc/yp-tools\* directory.   
  
Start the ypbind NIS client, and portmap daemons in the /etc/init.d directory and use  
the chkconfig command to ensure they start after the next reboot. Remember to use   
the rpcinfo command to ensure they are running correctly.   
  
[root@smallfry tmp]# service portmap start  
Starting portmapper: [ OK ]  
[root@smallfry tmp]# service ypbind start  
Binding to the NIS domain:  
Listening for an NIS domain server.  
[root@smallfry tmp]#   
  
[root@smallfry tmp]# chkconfig ypbind on  
[root@smallfry tmp]# chkconfig portmap on   
  
Note: Remember to use the rpcinfo -p localhost command to make sure they all started   
correctly.   
  
As the configuration examples refer to the NIS client and server by their hostnames,   
you'll have to make sure the names resolve correctly to IP addresses. This can be   
configured either in DNS, when the hosts reside in the same domain, or more simply   
by editing the /etc/hosts file on both Linux boxes.   
  
#  
# File: /etc/hosts (smallfry)  
#  
192.168.1.100 bigboy  
   
  
#  
# File: /etc/hosts (bigboy)  
#  
192.168.1.102 smallfry   
  
You can run the ypcat, ypmatch, and getent commands to make sure communication to the  
server is correct.   
  
[root@smallfry tmp]# ypcat passwd  
nisuser:$1$Cs2GMe6r$1hohkyG7ALrDLjH1:505:100::/home/nisuser:/bin/bash  
quotauser:!!:503:100::/home/quotauser:/bin/bash  
ftpinstall:$1$8WjAVtes$SnRh9S1w07sYkFNJwpRKa.:502:100::/:/bin/bash  
www:$1$DDCi/OPI$hwiTQ.L0XqYJUk09Bw.pJ/:504:100::/home/www:/bin/bash  
smallfry:$1$qHni9dnR$iKDs7gfyt..BS9Lry3DAq.:501:100::/:/bin/bash  
[root@smallfry tmp]#   
  
[root@smallfry tmp]# ypmatch nisuser passwd  
nisuser:$1$d6E2i79Q$wp3Eo0Qw9nFD/:504:100::/home/nisuser:/bin/bash  
[root@smallfry tmp]#   
  
[root@smallfry tmp]# getent passwd nisuser  
nisuser:$1$d6E2i79Q$wp3Eo0Qw9nFD/:504:100::/home/nisuser:/bin/bash  
[root@smallfry tmp]#   
  
Once your basic NIS functionality testing is complete, try to test a remote login.   
Failures in this area could be due to firewalls blocking TELNET or SSH access and   
the TELNET and SSH server process not being started on the clients.  
Logging In Via Telnet   
  
Try logging into the NIS client via telnet if it is enabled   
  
[root@bigboy tmp]# telnet 192.168.1.201  
Trying 192.168.1.201...  
Connected to 192.168.1.201.  
Escape character is '^]'.  
Red Hat Linux release 9 (Shrike)  
Kernel 2.4.20-6 on an i686  
login: nisuser  
Password:  
Last login: Sun Nov 16 22:03:51 from 192-168-1-100.simiya.com  
[nisuser@smallfry nisuser]$   
  
Logging In Via SSH   
  
Try logging into the NIS client via SSH.   
  
[root@bigboy tmp]# ssh -l nisuser 192.168.1.102  
nisuser@192.168.1.102  
[nisuser@smallfry nisuser]$   
  
In some versions of Linux, the NIS client's SSH daemon doesn't re-read the   
/etc/nsswitch.conf file you just modified until SSH is restarted. SSH logins,   
therefore, won't query the NIS server until this is done. Restart SSH on the NIS   
client.   
  
[root@smallfry root]# service sshd restart  
Stopping sshd:[ OK ]  
Starting sshd:[ OK ]  
[root@smallfry root]#  
  
  
======================================  
 Configure OpenLDAP Solaris:  
======================================  
======================================  
 Setting up the OpenLDAP server:  
======================================   
  
I won't go into too much detail here, as this part is fairly straight-forward.   
Basically, download and compile OpenLDAP 2.4.x with the options that you like,   
optionally create a package, and then install OpenLDAP.  
I used the following configure options:   
  
BDBDIR=/usr/local/BerkeleyDB.4.2 ; export BDBDIR  
LD\_LIBRARY\_PATH=${BDBDIR}/lib:/usr/sfw/lib \  
CPPFLAGS="-I${BDBDIR}/include/ -I/usr/sfw/include" \  
LDFLAGS="-L${BDBDIR}/lib -L/usr/sfw/lib" \  
./configure --with-tls=openssl --enable-overlays --enable-crypt \  
--enable-modules --enable-monitor --prefix=/opt/openldap \  
--enable-syslog --enable-proctitle --without-subdir   
  
make clean && make depend && make   
  
After installing OpenLDAP you will probably want to add some schemas. For solaris you  
need solaris.schema and I prefer to have my SUDO config in LDAP, so I also include   
it's schema:  
  
  
======================================  
 solaris.schema file:  
======================================   
  
# http://www.int-evry.fr/mci/user/procacci/ldap/solaris.schema  
#  
# solaris.schema  
# ''works in progress and incomplete''.  
# It would help if sun would publish this information!  
# If you have any comments/suggestion/correction  
# please let me know ( igor@ipass.net   
#  
# Some correction on oid and attributetype  
# were made by Marc Bourget ( bourget@up2.com )  
# Up2 Technologies (div. Teleglobe Communication Corp)  
# oid number and additional attributetype were taken from:  
# Solaris and LDAP Naming Service, Deploying LDAP in the Enterprise.  
# Tom Bialanski and Michael Haines, Sun Microsystems Press,  
# A Prentice Hall Title, 2001, ISBN 0-13-030678-9   
  
# Sun nisMapEntry attributes  
attributetype ( 1.3.6.1.1.1.1.28  
NAME 'nisPublickey'  
DESC 'nisPublickey'  
EQUALITY caseIgnoreIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.1.1.1.29  
NAME 'nisSecretkey'  
DESC 'nisSecretkey'  
EQUALITY caseIgnoreIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.1.1.1.12 SUP name  
NAME 'nisDomain' )   
  
# Sun additional attributes to RFC2307 attributes (NIS)  
attributetype ( 2.16.840.1.113730.3.1.30  
NAME 'mgrpRFC822MailMember'  
DESC 'mgrpRFC822MailMember'  
EQUALITY caseIgnoreIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
#attributetype ( 1.3.6.1.4.1.42.2.27.2.1.15  
# NAME 'rfc822MailMember'  
# DESC 'rfc822MailMember'  
# EQUALITY caseIgnoreIA5Match  
# SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.1.1.12  
NAME 'nisNetIdUser'  
DESC 'nisNetIdUser'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.1.1.13  
NAME 'nisNetIdGroup'  
DESC 'nisNetIdGroup'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.1.1.14  
NAME 'nisNetIdHost'  
DESC 'nisNetIdHost'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
# Sun NIS publickey objectclass  
objectclass ( 1.3.6.1.1.1.2.14  
NAME 'NisKeyObject'  
DESC 'NisKeyObject'  
SUP top  
MUST ( cn $ nisPublickey $ nisSecretkey )  
MAY ( uidNumber $ description ) )   
  
# Sun NIS domain objectclass  
objectclass ( 1.3.1.6.1.1.1.2.15  
NAME 'nisDomainObject'  
DESC 'nisDomainObject'  
SUP top AUXILIARY  
MUST ( nisDomain ) )   
  
# Sun NIS mailGroup objectclass  
objectclass ( 2.16.840.1.113730.3.2.4  
NAME 'mailGroup'  
DESC 'mailGroup'  
SUP top  
MUST ( mail )  
MAY ( cn $ mgrpRFC822MailMember ) )   
  
# Sun NIS nisMailAlias objectclass  
#objectclass ( 1.3.6.1.4.1.42.2.27.1.2.5  
# NAME 'nisMailAlias'  
# DESC 'nisMailAlias'  
# SUP top  
# MUST ( cn )  
# MAY ( rfc822mailMember ) )   
  
# Sun NIS nisNetId objectclass  
objectclass ( 1.3.6.1.4.1.42.2.27.1.2.6  
NAME 'nisNetId'  
DESC 'nisNetId'  
SUP top  
MUST ( cn )  
MAY ( nisNetIdUser $ nisNetIdGroup $ nisNetIdHost ) )   
  
# Below is optional unless you want to use ldap\_gen\_profile  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.15 SUP name  
NAME 'SolarisLDAPServers'  
DESC 'SolarisLDAPServers'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.16 SUP name  
NAME 'SolarisSearchBaseDN'  
DESC 'SolarisSearchBaseDN'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.17  
NAME 'SolarisCacheTTL'  
DESC 'SolarisCacheTTL'  
EQUALITY integerMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.18 SUP name  
NAME 'SolarisBindDN'  
DESC 'SolarisBindDN'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.19 SUP name  
NAME 'SolarisBindPassword'  
DESC 'SolarisBindPassword'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.20 SUP name  
NAME 'SolarisAuthMethod'  
DESC 'SolarisAuthMethod'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.21 SUP name  
NAME 'SolarisTransportSecurity'  
DESC 'SolarisTransportSecurity'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.24 SUP name  
NAME 'SolarisDataSearchDN'  
DESC 'SolarisDataSearchDN'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.25 SUP name  
NAME 'SolarisSearchScope'  
DESC 'SolarisSearchScope'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.26  
NAME 'SolarisSearchTimeLimit'  
DESC 'SolarisSearchTimeLimit'  
EQUALITY integerMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.27 SUP name  
NAME 'SolarisPreferedServer'  
DESC 'SolarisPreferedServer' )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.28 SUP name  
NAME 'SolarisPreferedServerOnly'  
DESC 'SolarisPreferedServerOnly'  
SINGLE-VALUE )   
  
attributetype ( 1.3.6.1.4.1.42.2.27.5.1.29 SUP name  
NAME 'SolarisSearchReferral'  
DESC 'SolarisSearchReferral'  
SINGLE-VALUE )   
  
objectclass ( 1.3.6.1.4.1.42.2.27.5.2.7  
NAME 'SolarisNamingProfile'  
DESC 'Solaris LDAP NSS Profile'  
SUP top STRUCTURAL  
MUST ( cn $ SolarisLDAPServers )  
MAY ( SolarisBindDN $ SolarisBindPassword $  
SolarisSearchBaseDN $ SolarisAuthMethod $  
SolarisTransportSecurity $ SolarisSearchReferral $  
SolarisDataSearchDN $ SolarisSearchScope $  
SolarisSearchTimeLimit $ SolarisCacheTTL ) )   
  
# End of solaris.schema  
  
  
======================================  
 sudo.schema file:  
======================================   
  
#  
# OpenLDAP schema file for Sudo  
# Save as /etc/openldap/schema/sudo.schema  
#   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.1  
NAME 'sudoUser'  
DESC 'User(s) who may run sudo'  
EQUALITY caseExactIA5Match  
SUBSTR caseExactIA5SubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.2  
NAME 'sudoHost'  
DESC 'Host(s) who may run sudo'  
EQUALITY caseExactIA5Match  
SUBSTR caseExactIA5SubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.3  
NAME 'sudoCommand'  
DESC 'Command(s) to be executed by sudo'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.4  
NAME 'sudoRunAs'  
DESC 'User(s) impersonated by sudo (deprecated)'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.5  
NAME 'sudoOption'  
DESC 'Options(s) followed by sudo'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.6  
NAME 'sudoRunAsUser'  
DESC 'User(s) impersonated by sudo'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
attributetype ( 1.3.6.1.4.1.15953.9.1.7  
NAME 'sudoRunAsGroup'  
DESC 'Group(s) impersonated by sudo'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )   
  
objectclass ( 1.3.6.1.4.1.15953.9.2.1 NAME 'sudoRole' SUP top STRUCTURAL  
DESC 'Sudoer Entries'  
MUST ( cn )  
MAY ( sudoUser $ sudoHost $ sudoCommand $ sudoRunAs $ sudoRunAsUser $ sudoRunAsGroup   
$ sudoOption $  
description )  
)   
  
These schema files should be installed in <openldap-dir>/etc/schemas/  
slapd.conf   
  
This is an example config for <openldap-dir>/etc/slapd.conf   
  
include /opt/openldap/etc/schema/core.schema  
include /opt/openldap/etc/schema/cosine.schema  
include /opt/openldap/etc/schema/nis.schema  
include /opt/openldap/etc/schema/inetorgperson.schema  
include /opt/openldap/etc/schema/solaris.schema  
include /opt/openldap/etc/schema/duaconf.schema  
include /opt/openldap/etc/schema/ppolicy.schema  
include /opt/openldap/etc/schema/sudo.schema   
  
# TLS Certificate  
TLSCACertificateFile /opt/openldap/etc/cacert.pem  
TLSCertificateFile /opt/openldap/etc/server..pem  
TLSCertificateKeyFile /opt/openldap/etc/server..pem  
TLSCipherSuite HIGH:MEDIUM:-SSLv2  
TLSVerifyClient allow  
#TLSVerifyClient demand | allow | never   
  
# ACL's  
access to dn.subtree="ou=People,dc=domain,dc=tld"   
attrs=userPassword,shadowLastChange  
by dn="cn=proxyagent,ou=profile,dc=domain,dc=tld" write  
by self write  
by anonymous auth  
by \* read   
  
# Do not allow users so change their uid/gid/groupmembership  
access to attrs=uid,uidNumber,gidNumber,memberUid  
by \* read   
  
access to dn.base=""  
by dn="cn=proxyagent,ou=profile,dc=domain,dc=tld" read  
by \* read   
  
access to dn.base="cn=Subschema"  
by anonymous none  
by \* read   
  
access to dn.subtree="ou=People,dc=domain,dc=tld"  
by self write  
by \* read   
  
access to dn.subtree="ou=Group,dc=domain,dc=tld"  
by \* read   
  
# Sudo rules are only readable by the dedicated sudoers account  
access to dn.subtree="ou=SUDOers,dc=domain,dc=tld"  
by dn="cn=sudoagent,ou=profile,dc=domain,dc=tld" read  
by \* none   
  
access to \*  
by \* read   
  
# MirrorMode Replication  
serverID 1   
  
database bdb  
suffix "dc=domain,dc=tld"  
rootdn "cn=Manager,dc=domain,dc=tld"   
  
#rootpw {SSHA}XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   
  
# Synchronisation/Replication  
overlay syncprov  
syncprov-checkpoint 100 10  
syncprov-sessionlog 100   
  
syncrepl rid=001  
provider=ldap://ldap2.domain.tld  
bindmethod=simple  
starttls=critical  
binddn="cn=proxyagent,ou=profile,dc=domain,dc=tld"  
credentials=secretpassword  
searchbase="dc=domain,dc=tld"  
schemachecking=on  
type=refreshAndPersist  
retry="60 +"   
  
# 2-Master mode  
mirrormode on   
  
# Indices to maintain   
  
index objectClass,uid,uidNumber,gidNumber,ou eq  
index cn,mail,surname,givenname eq,subinitial  
index memberUid eq  
index nisDomain eq  
index uniqueMember pres  
index sudoUser eq,sub   
  
# OVERLAY definitions: NEED TO BE \_\_AFTER\_\_ database definition they work on  
overlay ppolicy  
ppolicy\_default "cn=default,ou=policies,dc=domain,dc=tld"  
ppolicy\_hash\_cleartext on  
ppolicy\_use\_lockout   
  
overlay unique  
unique\_uri ldap:///ou=People,dc=domain,dc=tld?uidNumber,uid?sub  
unique\_uri ldap:///ou=Group,dc=domain,dc=tld?gidNumber,cn?sub   
  
# Performance tuning directives  
sizelimit 5000  
threads 16  
idletimeout 14400  
cachesize 10000  
checkpoint 256 15  
password-hash {SSHA}   
  
# Monitor  
database monitor  
access to dn.subtree="cn=Monitor"  
by dn="cn=Manager,dc=domain,dc=tld" write  
by users read  
by \* none  
  
  
======================================  
 Filling the LDAP Directory:  
======================================   
  
Next step is to fill the LDAP directory with some starting content  
Below you will find an example ldif file that can be used to jumpstart your LDAP   
directory. It creates a test user, group and people entries, a skeleton sudo   
infrastructure, configuration profiles and a password policy template.   
  
dn: dc=domain,dc=tld  
associatedDomain: domain.tld  
dc: ux  
objectClass: top  
objectClass: dcObject  
objectClass: domain  
objectClass: domainRelatedObject  
objectClass: nisDomainObject  
nisDomain: domain.tld  
o: Organisation Name   
  
dn: cn=Manager, dc=domain,dc=tld  
objectClass: organizationalRole  
cn: Manager   
  
dn: ou=profile, dc=domain,dc=tld  
ou: profile  
objectClass: top  
objectClass: organizationalUnit   
  
dn: ou=SUDOers, dc=domain,dc=tld  
ou: SUDOers  
objectClass: top  
objectClass: organizationalUnit   
  
dn: cn=defaults,ou=SUDOers, dc=domain,dc=tld  
objectClass: top  
objectClass: sudoRole  
description: Default sudoOptions go here  
sudoOption: ignore\_dot  
sudoOption: !mail\_no\_user  
sudoOption: root\_sudo  
sudoOption: log\_host  
sudoOption: logfile=/var/log/sudolog  
sudoOption: timestamp\_timeout=5  
cn: defaults   
  
dn: cn=Global\_Allowed\_NOPASS,ou=SUDOers, dc=domain,dc=tld  
sudoUser: ALL  
sudoCommand: /some/script.sh  
sudoHost: ALL  
objectClass: top  
objectClass: sudoRole  
sudoOption: !authenticate  
cn: Global\_Allowed\_NOPASS   
  
dn: ou=People, dc=domain,dc=tld  
ou: People  
objectClass: top  
objectClass: organizationalUnit   
  
dn: ou=Group, dc=domain,dc=tld  
ou: Group  
objectClass: top  
objectClass: organizationalUnit  
dn: cn=Users,ou=Group, dc=domain,dc=tld  
gidNumber: 1000  
objectClass: top  
objectClass: posixGroup  
cn: Users   
  
dn: cn=proxyagent,ou=profile, dc=domain,dc=tld  
userPassword:: MUNGED  
objectClass: top  
objectClass: person  
sn: proxyagent  
cn: proxyagent   
  
dn: cn=default,ou=profile, dc=domain,dc=tld  
defaultSearchBase: dc=domain,dc=tld  
authenticationMethod: simple  
followReferrals: TRUE  
profileTTL: 43200  
searchTimeLimit: 30  
objectClass: DUAConfigProfile  
defaultServerList: ldapserver1.domain.tld ldapserver2.domain.tld  
credentialLevel: proxy  
cn: default  
defaultSearchScope: one   
  
dn: cn=tls\_profile,ou=profile, dc=domain,dc=tld  
defaultSearchBase: dc=domain,dc=tld  
authenticationMethod: tls:simple  
followReferrals: FALSE  
bindTimeLimit: 10  
profileTTL: 43200  
searchTimeLimit: 30  
objectClass: top  
objectClass: DUAConfigProfile  
defaultServerList: ldapserver1.domain.tld ldapserver2.domain.tld  
credentialLevel: proxy  
cn: tls\_profile  
serviceSearchDescriptor: passwd: ou=People,dc=domain,dc=tld  
serviceSearchDescriptor: group: ou=Group,dc=domain,dc=tld  
serviceSearchDescriptor: shadow: ou=People,dc=domain,dc=tld  
serviceSearchDescriptor: netgroup: ou=netgroup,dc=domain,dc=tld  
serviceSearchDescriptor: sudoers: ou=SUDOers,dc=domain,dc=tld  
defaultSearchScope: one   
  
dn: ou=policies, dc=domain,dc=tld  
ou: policies  
objectClass: top  
objectClass: organizationalUnit   
  
dn: uid=testuser,ou=People, dc=domain,dc=tld  
shadowMin: 5  
sn: User  
userPassword:: MUNGED  
loginShell: /bin/bash  
uidNumber: 9999  
gidNumber: 1000  
shadowFlag: 0  
shadowExpire: -1  
shadowMax: 99999  
uid: testuser  
objectClass: top  
objectClass: person  
objectClass: organizationalPerson  
objectClass: posixAccount  
objectClass: shadowAccount  
gecos: Test User  
shadowLastChange: 0  
cn: Test User  
homeDirectory: /export/home/testuser  
shadowInactive: -1  
shadowWarning: 7   
  
dn: cn=default,ou=policies, dc=domain,dc=tld  
pwdFailureCountInterval: 30  
pwdSafeModify: FALSE  
pwdGraceAuthNLimit: 5  
pwdLockoutDuration: 10  
objectClass: pwdPolicy  
objectClass: person  
objectClass: top  
objectClass: pwdPolicyChecker  
pwdMaxFailure: 5  
pwdAllowUserChange: TRUE  
pwdMinLength: 5  
cn: default  
pwdAttribute: userPassword  
pwdMinAge: 5  
pwdLockout: TRUE  
pwdCheckQuality: 1  
pwdInHistory: 5  
sn: default policy  
pwdMustChange: FALSE  
pwdExpireWarning: 600  
pwdMaxAge: 10  
  
  
======================================  
 Configuring a Solaris 10 Client:  
======================================   
  
If you have defined a profile in your LDAP tree, it should be quite easy to setup a   
LDAP client on a Solaris 10 system.  
If you are using SSL or TLS with your server (you should), then you need to install   
the CA certificate first, so the server certificate can be checked.   
  
certutil -N -d /var/ldap  
certutil -A -d /var/ldap -n 'CA Name' -i /path/to/cacert.pem -a -t CT   
  
1. First copy /etc/nsswitch.ldap to /etc/nsswitch.ldap.bak and /etc/nsswitch to   
/etc/nsswitch.bak  
2. Edit /etc/nsswitch.ldap, making sure to change the entries for hosts and ipnodes   
to 'files dns'  
3. run ldapclient init:   
  
ldapclient init -v \  
-a proxyDN=cn=proxyagent,ou=profile,dc=domain,dc=tld \  
-a proxyPassword=secret \  
-a domainName=domain.tld \  
-a profileName=tls\_profile \  
ldapserver.domain.tld  
4. If all is well, LDAP should be configured now.   
  
Using listusers you should be able to see the ldap accounts in your userlist.  
  
  
======================================  
 Configuring PAM:  
======================================   
  
Next step is configuring pam to allow people to actually log-in using ldap accounts,   
and have their passwords stored in LDAP. Sun-SSH uses seperate pam names for each   
authentication method, and the sshd-pubkey method has it's own dedicated   
configuration.   
  
# pam.conf.ldapv2\_native\_client  
#  
# http://docs.sun.com/app/docs/doc/816-4556/6maort2te?a=view  
#  
# IMPORTANT NOTES from Gary Tay  
#  
# 1) This is a /etc/pam.conf with password management support that works for:  
#  
# Solaris10 Native LDAP Client  
# Solaris9 Native LDAP Client provided that:  
# - latest kernel patch and Patch 112960 are applied  
# - all the pam\_unix\_cred.so.1 lines are commented out  
# Solaris8 Native LDAP Client provided that:  
# - latest kernel patch and Patch 108993 are applied  
# - all the pam\_unix\_cred.so.1 lines are commented out  
#  
# 2) If modules for sshd or any are not defined, default is other  
# as seen by output of grep other /etc/pam.conf  
#  
# Notes from Mark Janssen  
#  
# 3) SSH Pubkey authentication needs its own pam rules on sshd-pubkey  
#  
# Authentication management  
#  
# login service (explicit because of pam\_dial\_auth)  
#  
login auth requisite pam\_authtok\_get.so.1  
login auth required pam\_dhkeys.so.1  
login auth required pam\_unix\_cred.so.1  
login auth required pam\_dial\_auth.so.1  
login auth binding pam\_unix\_auth.so.1 server\_policy  
login auth required pam\_ldap.so.1  
#  
# rlogin service (explicit because of pam\_rhost\_auth)  
#  
rlogin auth sufficient pam\_rhosts\_auth.so.1  
rlogin auth requisite pam\_authtok\_get.so.1  
rlogin auth required pam\_dhkeys.so.1  
rlogin auth required pam\_unix\_cred.so.1  
rlogin auth binding pam\_unix\_auth.so.1 server\_policy  
rlogin auth required pam\_ldap.so.1  
#  
# rsh service (explicit because of pam\_rhost\_auth,  
# and pam\_unix\_auth for meaningful pam\_setcred)  
#  
rsh auth sufficient pam\_rhosts\_auth.so.1  
rsh auth required pam\_unix\_cred.so.1  
rsh auth binding pam\_unix\_auth.so.1 server\_policy  
rsh auth required pam\_ldap.so.1  
#  
# PPP service (explicit because of pam\_dial\_auth)  
#  
ppp auth requisite pam\_authtok\_get.so.1  
ppp auth required pam\_dhkeys.so.1  
ppp auth required pam\_dial\_auth.so.1  
ppp auth binding pam\_unix\_auth.so.1 server\_policy  
ppp auth required pam\_ldap.so.1  
#  
# Default definitions for Authentication management  
# Used when service name is not explicitly mentioned for authentication  
#  
other auth requisite pam\_authtok\_get.so.1  
other auth required pam\_dhkeys.so.1  
other auth required pam\_unix\_cred.so.1  
other auth binding pam\_unix\_auth.so.1 server\_policy  
other auth required pam\_ldap.so.1  
#  
# passwd command (explicit because of a different authentication module)  
#  
passwd auth binding pam\_passwd\_auth.so.1 server\_policy  
passwd auth required pam\_ldap.so.1  
#  
# cron service (explicit because of non-usage of pam\_roles.so.1)  
#  
cron account required pam\_unix\_account.so.1  
#  
# Default definition for Account management  
# Used when service name is not explicitly mentioned for account management  
#  
other account requisite pam\_roles.so.1  
other account binding pam\_unix\_account.so.1 server\_policy  
other account required pam\_ldap.so.1  
#  
# Default definition for Session management  
# Used when service name is not explicitly mentioned for session management  
#  
other session required pam\_unix\_session.so.1  
#other session required pam\_mkhomedir.so.1  
#  
# Default definition for Password management  
# Used when service name is not explicitly mentioned for password management  
#  
other password required pam\_dhkeys.so.1  
other password requisite pam\_authtok\_get.so.1  
other password requisite pam\_authtok\_check.so.1  
other password required pam\_authtok\_store.so.1 debug server\_policy   
  
# Custom Stuff  
# Allow ssh-pubkey (SUN-SSH) logins to work  
sshd-pubkey account required pam\_unix\_account.so.1

======================================  
 Configuring a AIX 6.1 Client  
======================================  
   
  
Configuring AIX6.1 is quite easy, especially compared to Solaris.  
   
  
\* Make sure the LDAP client packages are installed  
 o idsldap.clt32bit61.rte 6.1.0.3 Directory Server - 32 bit Client  
 o idsldap.clt64bit61.rte 6.1.0.3 Directory Server - 64 bit Client  
 o idsldap.cltbase61.adt 6.1.0.3 Directory Server - Base Client  
 o idsldap.cltbase61.rte 6.1.0.3 Directory Server - Base Client  
 \* run: mksecldap -c -h ldapserver1,ldapserver2 -a   
 cn=proxyagent,ou=profile,dc=domain,dc=tld -p password -k   
 /etc/security/ldap/your-ca.kdb -w keydbpassword -A ldap\_auth  
 o Convert your cacert.pem file to a .kdb file using (java) gsk7ikm, and place it in   
 /etc/security/ldap/your-ca.kdb  
 o keydbpassword = the password you use in gsk7ikm to encrypt your keyring   
 (mandatory)  
 o password = the password used for the proxyagent  
  
======================================  
 Configuring a RHEL Client:  
======================================  
   
  
Configuring a Redhat Enterprise Linux Client is quite easy. It consists of the   
 following steps:  
   
  
\* Copy the CA-Certificate to /etc/openldap/cacerts/ca-cert.pem  
 \* Edit /etc/ldap.conf: Add the correct values for 'binddn' and 'bindpw'  
   
  
binddn cn=proxyagent,ou=profile,dc=domain,dc=tld  
 bindpw secret  
 \* Run /usr/bin/system-config-authentication  
 o Check 'Cache Information'  
 o Check 'Use LDAP', Check 'Use TLS' and fill in the ldap hostname and base-DN  
 o Check 'Use LDAP Authentication'  
 o Check 'Local authentication is sufficient'  
  
======================================  
 Configuring Netgroups:  
======================================  
   
  
Using the setup described above lets any ldap user with a valid account log in to any  
 ldap-enabled client machine. This might not be what you want. Using netgroups is a   
 method to limit ldap account visibility on a per system basis. Using netgroups you   
 can specify what (groups of) users can login and use what systems.  
 Configuring netgroups consists of the following steps:  
   
  
Import the following ldif-file into your directory:  
   
  
dn: ou=Netgroup, dc=domain,dc=tld  
 ou: netgroup  
 objectClass: top  
   
  
objectClass: organizationalUnit  
   
  
dn: cn=Admins, ou=Netgroup, dc=domain,dc=tld  
 objectClass: nisNetgroup  
 objectClass: top  
 nisNetgroupTriple: (,someuser,domain.tld)  
 cn: Admins  
   
  
dn: cn=App1, ou=Netgroup, dc=domain,dc=tld  
 objectClass: nisNetgroup  
 objectClass: top  
 nisNetgroupTriple: (,app1user,domain.tld)  
 memberNisNetgroup: Admins  
 cn: App1  
   
  
This example creates the Netgroup infrastructure, and populates it with 2 netgroups.   
 The 'App1' netgroup would be used on systems where 'App1' would run. The 'Admins'   
 netgroup is a group for the admins, and it's included in the 'App1' netgroup. This   
 way I only need to allow the App1 netgroup on that system, and it automatically   
 includes the users from the 'Admins' netgroup.  
  
 To specify a user in a netgroup, use a 'nisNetgroupTriple' where the value is: '(',   
 <hostname>, <username>, <domainname>, ')'. All fields are optional and can be left   
 out. In our case, we're mostly interested in the 'username' field, so the entries   
 look like '(,username,)'.  
 A netgroup can include another netgroup using 'memberNisNetgroup: netgroupname'.  
  
======================================  
 Solaris: Changing nsswitch.conf  
======================================  
   
  
We will be using the 'compat' support for netgroups, so we need to change the   
 'passwd' entry in /etc/nsswitch.conf from:  
   
  
passwd: files ldap  
   
  
to  
   
  
passwd: compat  
 passwd\_compat: ldap  
   
  
We are telling the nss system to use 'compat' (instead of the default files or ldap),  
 and telling it that the database that it should check for NIS entries is ldap   
 (default would be YP)  
  
======================================  
 AIX: Changing system settings for netgroups  
======================================  
   
  
For AIX the following changes need to be made to enable netgroups:  
   
  
\* In /usr/lib/security/methods.cfg, change the LDAP group, add the options line:  
   
  
LDAP:  
 program = /usr/lib/security/LDAP  
 program\_64 =/usr/lib/security/LDAP64  
 options = netgroup  
 \* In /etc/group, add a line at the end:  
   
  
+:  
 \* In /etc/security/user, change the default group:  
   
  
SYSTEM = compat  
  
======================================  
 Allowing netgroups  
======================================  
   
  
Every netgroup you want to allow on the system needs to be included in the   
 /etc/passwd file. Make sure you use the correct format, otherwise you will not be   
 able to login. For Solaris this format needs to be:  
   
  
+@netgroupname:x:::::  
 +@othernetgroup:x:::::  
   
  
If you only add '+@netgroupname' things seem to work, you can see the accounts with   
 'listusers' and even 'su' to them, however you still can't login with these   
 accounts. If you add the entry as specified above, and then run 'pwconv' the entry   
 will be copied to '/etc/shadow' in the correct format and you should then be able   
 to login with netgroup-listed accounts.  
 For AIX you can just specify the simpler:  
   
  
+@netgroupname  
 +@othernetgroup  
   
  
It's recomendable to create dedicated netgroups for any system or group of systems   
 that have their own user limitations. It's also a good idea to include the 'admin'   
 netgroup in any netgroup you create or explicitly include it on every system.  
  
======================================  
 Creating home directories  
======================================  
   
  
Linux and AIX have PAM modules to create a home directory for a user if one doesn't   
 exist. Solaris sadly doesn't have a PAM module for this (and I couldn't get the   
 linux module working for solaris).  
 The Linux PAM module is pam\_mkhomedir. You can include it in your PAM stack as   
 follows:  
   
  
session required pam\_mkhomedir.so skel=/etc/skel/ umask=0022  
   
  
The AIX PAM module is called pam\_mkuserhome, however, I have not been able to get it   
 to create an actual directory in my experiments. Since I already need to have a   
 work-around for Solaris I used this method for AIX as well.  
   
  
\* Create a mkhome script and put it in /usr/local/bin  
   
  
#!/bin/sh  
   
  
if [ -d ${HOME} ]; then  
 exit 0  
 fi  
   
  
mkdir -p ${HOME}  
 cp -r /etc/skel/.???\* ${HOME}  
 cp -r /etc/skel/\* ${HOME}  
 chown ${SUDO\_UID}:${SUDO\_GID} ${HOME} ${HOME}/\* ${HOME}/.???\*  
 echo "Created ${HOME}"  
 exit 0  
 \* Allow this script to be run using sudo, without prompting for a password  
   
  
dn: cn=Global\_Allowed\_NOPASS,ou=SUDOers, dc=domain,dc=tld  
 sudoUser: ALL  
 sudoCommand: /usr/local/bin/mkhome  
 sudoHost: ALL  
 objectClass: top  
 objectClass: sudoRole  
 sudoOption: !authenticate  
 cn: Global\_Allowed\_NOPASS  
 \* Call sudo /usr/local/bin/mkhome from /etc/profile when a home directory can't be   
 found  
   
  
if [ ! -d $HOME ]  
 then  
 /usr/bin/sudo /usr/local/bin/mkhome  
 cd $HOME  
 fi  
  
======================================  
 MAIL ///////////////////////////////  
======================================  
  
  
mail is a command line e-mail client for Unix and Unix-like operating systems.  
  
   
Example usage  
  
mail -s "You've got mail" -c cc.rider@b.c   
somebody@example.com   
   
anotherbody@example.net   
   
  
This sends a message with the subject "You've got mail" to two recipients, somebody   
and anotherbody, and CCs (copies) a third, cc.rider. The message will be typed   
after the command is entered and will be ended with Control-D.  
  
If you want to send a message in one line, use any Unix command sequence that   
generates text. For example:  
  
  
o "Some message" | mail -s "meeting today"   
somebody@example.com   
   
  
This is especially useful for having a system report its status automatically through  
E-mail.  
  
  
  
There is also a -a option for using an ISO-8859 character set beyond US Ascii. For   
instance:  
  
  
  
mail -s "You've got mail" -a ISO-8859-8   
somebody@example.com   
   
  
The -a ISO-8859-8 option tells Mail that the message may include Hebrew characters.  
  
  
  
======================================  
MAILX ///////////////////////////////  
======================================  
  
mailx is a Unix utility program for sending and receiving mail, also known as a Mail   
 User Agent program. It is an improved version of the mail utility.  
  
  
  
mailx is a lightweight mail program which has a command syntax similar to ed. Mailx   
 allows one to send and read email. Mailx cannot, by itself, receive email from   
 another computer. It reads messages from a file on the local machine, which are   
 delivered there by a local delivery agent such as procmail.  
  
  
  
A simplified syntax with the most commonly used options is:  
mailx [-s subject] [-a attachment ] [-r from-addr] to-addr . . .  
  
 \* -s subject of email (could be inserted later)  
 \* -r indicates the email's sender (not a standard argument)  
 \* -a file to be attached to email (in some versions)  
 \* -a specify additional header fields (in other versions)  
  
  
  
The end of message is indicated by a single '.' or by hitting ctrl+d. In the simple   
 send usage, you just type your message directly into mailx. But in real life,   
 you'll decide to edit the message after you've been typing for a while. Mailx   
 interprets input lines beginning with a tilde (~) as commands. Its ~v command   
 causes mailx to invoke the text editor of your choice (defined by the VISUAL   
 environment variable) on the message in progress, saved in a temporary file. It can  
 be argued this feature makes Mailx a more powerful email composing tool than   
 typical Graphical User Interface (GUI) Mail User Agents.  
  
   
Example usage  
  
A simple example  
  
 $ mailx -s "From mailx" abc@cde.com   
 type the body   
 ...   
 EOT (Ctrl+d)   
 $   
   
  
  
Simple syntax to send email with subject 'From mailx' to abc@cde.com.  
  
A more complex example:  
  
 $ mailx -s "the subject" -a arg1 -a arg2 -a "From:me<email@mail.com>"  
 "person1<person1@hotmail.com>" "person2<person2@hotmail.com>"   
 message body   
 ...   
 EOT   
 $  
   
  
Sends message to person1 and person 2 with arq1 and arq2 as anex. The subject of   
message will be "the subject" and the receivers will see "me" as the sender.  
  
Process the Variable in the Body of Mail example:  
  
 $ mailx -s "Subject" "dhil@yahoo.com" <<EOT   
 hi Dream,   
 `date` this is the date on unix now   
 EOT   
 $   
  
In the above Example it will send the mail with the current date processed as follows. This  
is actually an example of a unix shell feature called a "here document." The mailx command   
has no idea where the text is coming from, it's just reading lines from its standard input.  
The stuff from the double less-than through the second EOT is interpreted by the shell,  
which runs the date command and inserts its output in the appropriate place.  
  
  
 hi Dream,   
 Thu Aug 23 02:25:38 EDT 2007 this is the date on unix now   
  
One line example as a reminder for job completion  
  
Suppose you have a job which it is going to run for a while. You don't want to stare at the  
screen to wait for its completion. You can switch to other terminal to work on other things  
but want to get notified the minute the job is completed. Here's the trick to do so:  
  
 sleep 24 && echo | mailx -s "XXX job completed" xxxxx@gmail.com   
  
here "sleep 24" is just an example test run command to count for 24 seconds before the mailx  
command is executed. You should replace it with the job submission command of your choice.  
Once the job is completed, a mail with title "XXX job completed" and empty body will be sent  
to xxxxx@gmail.com.

======================================  
 Sendmail   
======================================  
  
Download Sendmail  
  
Download Sendmail[1]. You may be automatically offered a short initial message which will  
indicate the current release. These instructions below assume version 8.10.0 or later.  
  
Build and install Sendmail for your machine. In most cases, this consists of unpacking the  
distribution, reading the README and sendmail/README files, and typing Build in the Sendmail   
directory. See the INSTALL file in the distribution's top-level directory for details.  
  
Set up Sendmail  
  
Understand that Sendmail uses information from the Domain Name System (DNS) to figure out  
which IP addresses go with which mailboxes.  
  
Choose an available domain name. In our example, we will use yourdomain.com.  
  
Configure your DNS on the server. Establish two machines as primary and secondary name  
servers for your domain. Knowledge of how to do this is assumed; otherwise, read the O'Reilly   
book "DNS and BIND", 4th Edition is highly recommended. Familiarize yourself with BIND before  
continuing.  
  
Configure MX records for your domain (Note: CNAME records can not be used; see § 5.2.2 of RFC  
1123 for details.) MX records are explained in the O'Reilly Sendmail book; the 2nd edition gives   
an overview in § 15.3 and describes how to configure them in § 21.3, whereas the third  
edition explains everything about them in § 9.3. You have two options for MX records:  
  
Determine your connection method: If the mail server which will serve your new domain has a  
full-time connection to the Internet, it should be the primary MX host for your domain. In  
this configuration, your MX records would look like this: yourdomain.com. IN MX 10  
yourmailserver.yourdomain.com.  
  
  
Otherwise, you will need to find another machine to queue mail for your domain when you are  
not connected. Be sure to get the machine owners' approval first. That machine must be  
configured to allow relaying to your domain. If it is running Sendmail, this can be as simple  
as adding your domain to the relay-domains file on that machine. You would then point your MX  
records at that machine. For example:  
  
yourdomain.com. IN MX 10 yourmailserver.yourdomain.com.  
  
yourdomain.com. IN MX 20 othermailserver.otherdomain.com.  
  
  
Configure Sendmail  
  
Read the cf/README file thoroughly. It will give you instructions on creating a .mc file in  
the cf/cf directory. Your mailserver.mc file will typically look something like: divert(-1)dnl  
  
#  
  
# This file contains the global definitions for yourdomain.com  
  
#  
  
divert(0)dnl  
  
VERSIONID(`@(#)mailserver.mc 1.0 (yourdomain.com) 5/1/97')  
  
OSTYPE(solaris2)dnl  
  
DOMAIN(yourdomain.com)dnl  
  
FEATURE(`virtusertable', `dbm /etc/mail/virtusertable')dnl  
  
MAILER(local)dnl  
  
MAILER(smtp)dnl  
  
Your actual OS will be substituted for solaris2.  
  
  
A typical cf/domain/yourdomain.com.m4 file that looks something like: divert(-1)dnl  
  
#  
  
# This file contains the global definitions for yourdomain.com  
  
#  
  
divert(0)dnl  
  
VERSIONID(`@(#)yourdomain.com.m4 1.0 (yourdomain.com) 5/1/97')  
  
FEATURE(`use\_cw\_file')dnl  
  
  
It may have some other feature()'s and define()'s as well. The virtual user table is the key  
to all of this.  
  
Generate your /etc/mail/sendmail.cf file from your mailserver.mc file, so type: cd sendmail-  
VERSION/cf/cf  
  
/Build mailserver.cf  
  
cp mailserver.cf /etc/mail/sendmail.cf  
  
  
Create the virtual user table. This is explained in detail in the Sendmail book: § 19.6.28 of  
the 2nd edition, or § 4.8.51 of the 3rd edition; an overview is given here. The table is a  
database that maps virtual addresses into real addresses. You create a text file where each  
line has a key/value pair, separated by a TAB. For example: Example 1: joe@yourdomain.com  
jschmoe  
  
jane@yourdomain.com jdoe@othercompany.com  
  
@yourdomain.com jschmoe In this first example, the address joe@yourdomain.com will be mapped  
to the local user jschmoe; jane@yourdomain.com will be mapped to the remote user  
jdoe@othercompany.com, and anything else coming in to yourdomain.com will also go to jschmoe.  
  
  
Example 2: joe@yourdomain.com jschmoe%3  
  
bogus@yourdomain.com &npsp; error:nouser No such user here  
  
list@yourdomain.com yourdomain-list  
  
@yourdomain.com %1@othercompany.com In this second example, the address joe@yourdomain.com  
will be mapped to the local user jschmoe%3 (see note 3 below for an explanation of what the  
%3 means), the address bogus@yourdomain.com will return the indicated error, the address  
list@yourdomain.com will be mapped to the local user yourdomain-list (which you would use the  
aliases file to ultimately resolve) and every other user at yourdomain.com will be mapped to  
a remote user of the same name at othercompany.com.  
  
  
  
If you have a local user, say sam, and there is no key for sam@yourdomain.com and no catch-  
all key for @yourdomain.com, then Sendmail will fall back to the local user sam when  
resolving sam@yourdomain.com. To prevent this, you must use either a catch-all key or an  
explicit key for sam@yourdomain.com; the error:nouser example above may be useful in this  
instance.  
  
If you want a virtual address to resolve to more than one real address, you need to do it  
indirectly. Have the virtual address resolve to a local alias, then have the local alias  
resolve to the desired set of addresses. For example, in the virtual user table:  
joe@yourdomain.com localjoe  
  
then in the aliases file:  
  
  
localjoe: joe@othercompany.com, jane@othercompany.com  
  
In the above example: joe@yourdomain.com jschmoe%3  
  
  
The %3 is the preservation of the optional +detail part of the original address. In general,  
+detail means that when Sendmail gets an address like user+detail@domain, then if domain is  
in class w (see step 7 below), sendmail checks to see if user+detail can be resolved, then  
falls back to just plain user if not. Thus all of: joe@yourdomain.com  
  
joe+foo@yourdomain.com  
  
joe+reallylongextrapart@yourdomain.com  
  
would all match the above entry, with %3 preserving the +detail part of nothing, +foo and  
+reallylongextrapart respectively.  
  
  
Multiple domains are allowed, and virtual addresses in each domain are independent. So for  
example, you could have: joe@yourdomain1.com localjoe  
  
joe@yourdomain2.com joe@othercompany.com  
  
joe@yourdomain3.com localjoe  
  
joe@yourdomain4.com error:nouser No such user here  
  
  
For people administering multiple domains, it may be easier to keep each domain's list in a  
separate file, then write a short script to concatenate all such files together into a master  
virtual user table. But we're getting ahead of ourselves; that's the next step...  
  
  
Now the name servers are setup, register your domain using one of the registries. As you  
register, inform the registry of the two name servers, and then the domain will point to your  
server.  
  
  
Build the Sendmail User Table  
  
Build the virtual user table. If the above virtual user table text file is located at  
sourcefile, and you are using the dbm database type, then use the command: makemap dbm  
/etc/mail/virtusertable < sourcefile  
  
Note: if you built Sendmail with NEWDB instead of NDBM, then use hash instead of dbm in the  
above line.  
  
This creates one or more non-text files (typically /etc/mail/virtusertable.dir and  
/etc/mail/virtusertable.pag, or /etc/mail/virtusertable.db), but does not actually change  
/etc/mail/virtusertable itself, so this is the recommended location for sourcefile.  
  
If you would like to reverse-map local users for out-bound mail, you will need to add support  
for the generics table to your .mc file: FEATURE(`genericstable', `dbm  
/etc/mail/genericstable')dnl  
  
GENERICS\_DOMAIN\_FILE(`/etc/mail/generics-domains')dnl  
  
  
And you will need to create /etc/mail/genericstable which is like /etc/mail/virtusertable  
above except the columns are reversed: jschmoe joe@yourdomain.com  
  
Add your domain names to Sendmail  
  
Add each new domain name to sendmail's class w. This is typically done by adding a line to  
/etc/mail/local-host-names (known as /etc/sendmail.cw prior to version 8.10) with the value  
of each domain name. Likewise, if you are using the genericstable, you should add any domains  
you wish to reverse-map to /etc/mail/generics-domains.  
  
Restart or SIGHUP sendmail.  
  
You do not need to restart sendmail when changing the virtual user or generics tables, only  
when changing /etc/mail/sendmail.cf or class files such as /etc/mail/local-host-names. An  
extra step is required for hosts not connected full-time. As noted in the MX configuration  
section, if you use another host to queue your mail until you connect, you will have to force  
delivery of mail queued on the secondary mail server. To accomplish this, when your primary  
server connects, you should run the script etrn.pl which comes in the contrib directory of the   
sendmail distribution: etrn.pl secondary-mx-host yourdomain.com  
  
It may be advisable to put this at the end of the Sendmail start-up script on any primary MX.  
It would be especially useful as a follow-up to whatever script initiates the connection on  
primary MXs without full-time connections.  
  
At this point, you should be set, and people should be able to send e-mail to addresses  
@yourdomain.com.  
  
  
 \* Note: if you built Sendmail with NEWDB instead of NDBM, then use hash   
 instead of dbm in the above line.  
 \* This creates one or more non-text files (typically   
 /etc/mail/virtusertable.dir and /etc/mail/virtusertable.pag, or   
 /etc/mail/virtusertable.db), but does not actually change   
 /etc/mail/virtusertable itself, so this is the recommended location for   
 sourcefile.  
 \* If you would like to reverse-map local users for out-bound mail, you will  
 need to add support for the generics table to your .mc file:  
 o FEATURE(`genericstable', `dbm /etc/mail/genericstable')dnl  
 o GENERICS\_DOMAIN\_FILE(`/etc/mail/generics-domains')dnl  
 \* And you will need to create /etc/mail/genericstable which is like   
 /etc/mail/virtusertable above except the columns are reversed:  
 o jschmoe joe@yourdomain.com   
  
Add your domain names to Sendmail  
  
 1. Add each new domain name to sendmail's class w. This is typically done by   
 adding a line to /etc/mail/local-host-names (known as /etc/sendmail.cw prior to   
 version 8.10) with the value of each domain name. Likewise, if you are using the  
 genericstable, you should add any domains you wish to reverse-map to   
 /etc/mail/generics-domains.  
 2. Restart or SIGHUP sendmail.  
 3. You do not need to restart sendmail when changing the virtual user or generics   
 tables, only when changing /etc/mail/sendmail.cf or class files such as   
 /etc/mail/local-host-names.  
 \* An extra step is required for hosts not connected full-time. As noted in   
 the MX configuration section, if you use another host to queue your mail   
 until you connect, you will have to force delivery of mail queued on the   
 secondary mail server. To accomplish this, when your primary server   
 connects, you should run the script etrn.pl which comes in the contrib   
 directory of the sendmail distribution:  
 o etrn.pl secondary-mx-host yourdomain.com  
 o It may be advisable to put this at the end of the Sendmail start-up  
 script on any primary MX. It would be especially useful as a   
 follow-up to whatever script initiates the connection on primary   
 MXs without full-time connections.  
 o At this point, you should be set, and people should be able to send  
 e-mail to addresses @yourdomain.com.  
  
Test your configuration file  
  
 1. Test your configuration and make sure everything works as expected before   
 announcing the new domain name and mail addresses for that domain. If things   
 don't work as expected, you can test with Sendmail's test mode:  
 \* sendmail -bt  
 \* Here are some examples of things to try in test mode (make sure the   
 domain is in class w:):  
 o $=w  
 o # is the map working?  
 o /map virtuser joe@yourdomain.com   
 o /map virtuser jane@yourdomain.com   
 o /map virtuser @yourdomain.com  
 o is the rewriting working? #\*\* ,0 joe@yourdomain.com #\*\* , 0 some@yourdomain.com   
  
   
Tips  
  
 \* What Sendmail does:  
 o Listen on network ports for mail.  
 o Sort mail and deliver it locally or externally to other servers.  
 o Append mail to files or pipe it through other programs.  
 o Queue mail (if immediate delivery fails).  
 o Convert email addresses to/from user names, or handle mailing lists.  
 o Reads rules for special mail handling, so it can try to catch spam, or   
 check for correctness.  
 \* If you built Sendmail with NEWDB instead of NDBM, you will have to use hash   
 instead of dbm in the above line.  
  
   
  
   
======================================  
MTA  
======================================  
  
A mail transfer agent (MTA) (also called a mail transport agent, message transfer agent, or  
smtpd (short for SMTP daemon), is a computer program or software agent that transfers  
electronic mail messages from one computer to another.  
  
The term mail server is also used to mean a computer acting as an MTA that is running the  
appropriate software. The term mail exchanger (MX), in the context of the Domain Name System  
formally refers to an IP address assigned to a device hosting a mail server, and by extension  
also indicates the server itself.  
  
  
Overview  
  
An MTA receives mail from another MTA (relaying) or from a mail user agent (MUA). The MTA  
works behind the scenes, while the user usually interacts with the MUA. Every time an MTA  
receives an e-mail, it will add a "Received:" trace header field to the top of the message.  
In this way, there is a record of which MTAs handled the e-mail and in which order. Upon  
final delivery, the "Return-Path:" header will also be added to record the return path.  
  
The delivery of e-mail to a user's mailbox typically takes place via a mail delivery agent  
(MDA); many MTAs have basic MDA functionality built in, but a dedicated MDA like procmail can  
provide more sophisticated functionality.  
  
According to one survey, sendmail, Microsoft Exchange Server, Postfix, and Exim together  
control over 85% of market share for SMTP service.[citation needed]  
  
Another survey suggests a more balanced playing field, though it included hosted e-mail  
services such as Postini.[1]  
  
See also

* [MX record](http://en.wikipedia.org/wiki/MX_record)
* [List of mail servers](http://en.wikipedia.org/wiki/List_of_mail_servers)
* [Comparison of mail servers](http://en.wikipedia.org/wiki/Comparison_of_mail_servers)
* [Mail user agent](http://en.wikipedia.org/wiki/Mail_user_agent)
* [Mail delivery agent](http://en.wikipedia.org/wiki/Mail_delivery_agent)
* [SMTP proxy](http://en.wikipedia.org/wiki/SMTP_proxy)

======================================  
 SECURITY /////////////////////////  
======================================  
  
  
======================================  
 NAT:  
======================================  
   
  
Network Address Translation (NAT):  
   
  
An individual on a computer on the private network may point their web browser to a   
 site on the internet. This request is recognized to be beyond the local network so   
 it is routed to the Linux gateway using the private network address. The request   
 for the web page is sent to the web site using the external internet IP address of   
 the gateway. The request is returned to the gateway which then translates the IP   
 address to computer on the private network which made the request. This is often   
 called IP masquerading. The software interface which enables one to configure the   
 kernel for masquerading is iptables (Linux kernel 2.4) or ipchains (Linux kernel   
 2.2)  
   
  
The gateway computer will need two IP addresses and network connections, one to the   
 private internal network and another to the external public internet.  
   
  
A note on private network IP addresses: A set of IP addresses has been reserved by   
 IANA for private networks. They range from 192.168.0.1 to 192.168.254.254 for a   
 typical small business or home network and are often referred to as CIDR private   
 network addresses. Most private networks conform to this scheme.

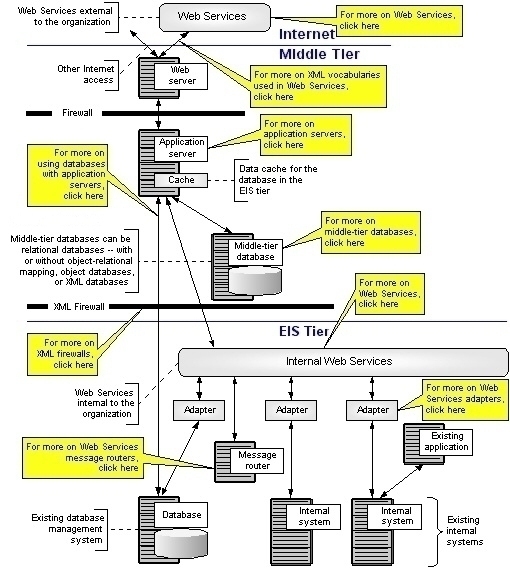
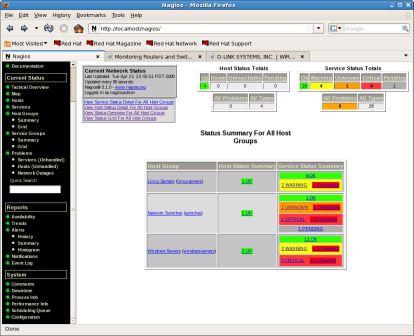
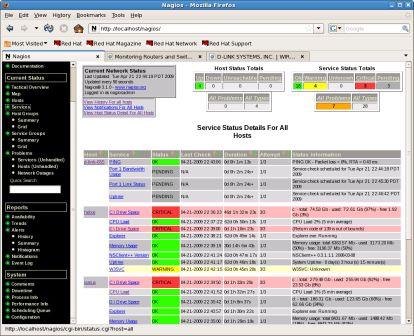
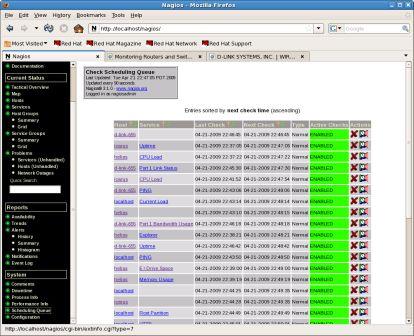
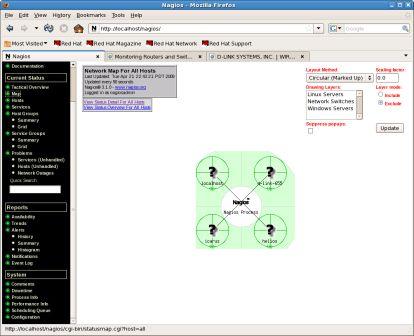
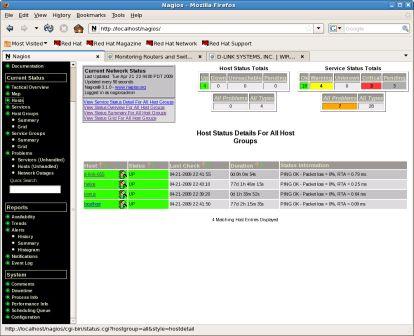
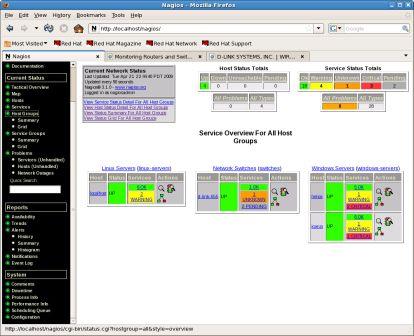
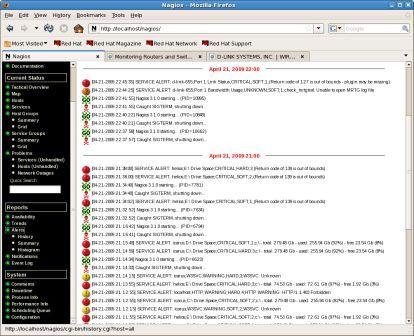
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block**  24 bit block in class A 20 bit block in class B 16 bit block in class C | **Range**  10.0.0.0-10.255.255.255 172.16.0.0-172.31.255.255 192.168.0.0-192.168.255.255 | **CIDR Notation**  10.0.0.0/8 172.16.0.0/12 92.168.0.0/16 | **Default Subnet Mask**  255.0.0.0 255.240.0.0 255.255.0.0 | **Number of hosts**  16,777,216 1,048,576 65,536 |

The actual number of hosts will be fewer than listed because addresses on each subnet  
 will be reserved as a broadcast address, etc.  
   
  
This is detailed in RFC 1918 - Address Allocation for Private Internets. For a   
 description of class A, B, and C networks see the YoLinux Networking Tutorial class  
 description.  
   
  
The private networks may be subdivided into various subnets as desired.  
   
  
Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Range**  10.2.3.0-10.2.4.255  172.16.0.0-172.17.255.255  192.168.5.128-192.168.5.255 | **CIDR Notation**  10.2.3.0/23  172.16.0.0/15  192.168.5.128/25 | **Default Subnet Mask**  255.255.254.0  255.254.0.0  255.255.255.128 | **Number of hosts**  512  132608  128 |

CertGuide.com: Network Subnets  
======================================  
 SELINUX:  
======================================  
   
  
A Linux kernel integrating SELinux enforces mandatory access control policies that   
 confine user programs and system servers to the minimum amount of privilege they   
 require to do their jobs. This reduces or eliminates the ability of these programs   
 and daemons to cause harm when compromised (via buffer overflows or   
 misconfigurations, for example). This confinement mechanism operates independently   
 of the traditional Linux access control mechanisms. It has no concept of a "root"   
 super-user, and does not share the well-known shortcomings of the traditional Linux  
 security mechanisms (such as a dependence on setuid/setgid binaries).  
   
  
The security of an unmodified Linux system depends on the correctness of the kernel,   
 all the privileged applications, and each of their configurations. A problem in any  
 one of these areas may allow the compromise of the entire system. In contrast, the   
 security of a modified system based on an SELinux kernel depends primarily on the   
 correctness of the kernel and its security policy configuration. While problems   
 with the correctness or configuration of applications may allow the limited   
 compromise of individual user programs and system daemons, they do not pose a   
 threat to the security of other user programs and system daemons or to the security  
 of the system as a whole.  
   
  
From a puristic perspective, SELinux provides a hybrid of concepts and capabilities   
 drawn from mandatory access controls, mandatory integrity controls, role-based   
 access control (RBAC), and type enforcement architecture. Third-party tools enable   
 one to build a variety of security policies.  
   
  
To determine if you're running the targeted policy, verify the following:  
   
  
\* The file /etc/selinux/config should contain the line SELINUXTYPE=targeted, but note  
 that this instructs the computer which policy to use at boot. If you change the   
 machine between different policies, leave the SELINUXTYPE variable with a different  
 value than the running policy before you reboot.  
 \* Running the id command should return something similar to  
   
  
uid=0(root) gid=0(root)   
 groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)   
 context=root:system\_r:unconfined\_t  
   
  
The final part of root's security context tells us that the root shell is running in   
 the unconfined\_t domain, indicating that the targeted policy is in use. On a system  
 running the strict policy a root shell will have the SELinux context of either   
 root:staff\_r:staff\_t or root:sysadm\_r:sysadm\_t. You can also run the id -Z command   
 to see your security context without the Unix UID/GID information (useful for shell  
 scripts).  
   
  
The daemons that have policy written for them will run in their own domains by   
 default, but the administrator may configure them to run in the domain unconfined\_t  
 by specifying that they should not have a domain transition when executed. For   
 example the command setsebool -P httpd\_disable\_trans 0 causes the httpd process to   
 run in domain unconfined\_t. Every daemon has a boolean to cause it to run in the   
 unconfined\_t domain. This can be used if the administrator is unable to get it   
 working correctly in its own domain (although it is recommended that for best   
 security the policy be modified if necessary to permit the daemon in question to   
 run in a restrictive domain). Running daemons in the unconfined\_t domain in this   
 manner reduces the security of the system and should be avoided if possible.  
   
  
Changing the values of booleans can be done with the setsebool command or with the   
 program system-config-securitylevel. When using setsebool make sure you use the -P   
 option if you want the change to be preserved across reboots.  
   
  
To view the values of booleans use the getsebool command. To retrieve the value of a   
 single bool specify the name on the command line, for example getsebool   
 httpd\_disable\_trans. To view the values of all booleans use the getsebool -a   
 command.  
   
  
The easiest way of changing the booleans is through the system-config-securitylevel   
 command as shown in Figure 1, Change the values of booleans. The httpd server has   
 more booleans than most daemons because it is very configurable, and the   
 configuration of the SELinux policy needs to match the configuration of the   
 daemon.  
   
  
Probably the most commonly used booleans in the targeted policy will be the ones to   
 disable the SELinux protection for daemons that have been configured in a way that   
 does not work well with SELinux. We don't recommend that you enable such booleans.   
 They are only provided as emergency measures. If business requirements force you to  
 run a daemon in a way that SELinux can't restrict, disabling the protection for   
 that daemon is much better than disabling it for the entire system.  
   
  
The policy files for the daemons when using the targeted policy is located in the   
 /etc/selinux/targeted/src/policy/domains/program/ directory. The policy source   
 files are commonly known as .te files which represent the naming convention such as  
 syslogd.te.  
   
  
A policy, or .te file, contains the rules for the associated domain. The syslogd.te   
 file, for instance, defines the rules for the operation of the domain syslogd\_t   
 including operations such as logging to the console, modification and creation of   
 log files, and remote logging, to name a few.  
   
  
The policy file must match the file contexts, or .fc file located in   
 /etc/selinux/targeted/src/policy/file\_contexts/program/. File contexts files list   
 the security contexts which must be applied to files and directories that the   
 daemon uses. For example, the file named.fc contains:  
   
  
/var/named(/.\*)? system\_u:object\_r:named\_zone\_t  
 /var/named/data(/.\*)? system\_u:object\_r:named\_cache\_t  
   
  
The first line tells us that the /var/named/ directory is of the type named\_zone\_t.   
 The second line tells us that the /var/named/data/ directory has the type   
 named\_cache\_t.  
   
  
/usr/sbin/named -- system\_u:object\_r:named\_exec\_t  
   
  
Tells us that the named executable is of type named\_exec\_t. The naming convention for  
 daemon entry point executables is X\_exec\_t where X is the name of the daemon   
 domain.  
   
  
This causes a transition from the domain unconfined\_t to the daemon domain (named\_t   
 in this example) when the daemon is executed. When using the strict policy daemons   
 have to be started from an administrative session (role sysadm\_r and domain   
 sysadm\_t) for correct operation. With the targeted policy, this is not an issue as   
 unconfined\_t is the only domain used for user logins (either administrator or a   
 regular user).  
   
  
The main policy file for named is named.te which contains the rules to permit access   
 to the named\_t domain as well as to define the domain and cause transitions to it.   
 The most significant line in named.te is:  
   
  
daemon\_domain(named, `, nscd\_client\_domain')  
   
  
This defines the domain named\_t and permits it to perform all the basic operations   
 that daemons perform such as writing a pid file to /var/run, forking a child   
 process, logging to syslog, etc. It also has policy to cause an automatic domain   
 transition from unconfined\_t to named\_t when an executable of type named\_exec\_t is   
 executed.  
 Objectives of the Targeted Policy  
   
  
The targeted policy was developed because the strict policy was considered to be too   
 difficult for many system administrators to manage. When SELinux was initially   
 introduced into Fedora Core 2, there was some negative feedback about the ease of   
 use. With the release of Fedora Core 3, the targeted policy was the default, and   
 there were very few complaints. Common estimates suggest that at least two million   
 people are using Fedora Core 3 without even realizing that they are using SELinux.   
 Their machine does what they want it to do, and they don't notice that daemons are   
 not permitted to perform certain operations those operations are not performed by   
 daemons in the normal operation of a system with a typical configuration.  
   
  
The general aim of policy development is that the strict policy will become easier to  
 configure and better tuned for default configurations while the targeted policy   
 will get an increasing number of targets to support more daemons. Through these   
 developments the targeted and strict policies can be considered to be converging as  
 the strict policy becomes easier to use and the targeted policy becomes more   
 strict. But it seems unlikely that we will be able to merge the policies in the   
 foreseeable future. The fundamental difference between strict and targeted is that   
 strict uses the identity and role features of SELinux while targeted does not.  
   
  
Some of the usability features of the targeted policy derive from many of the daemons  
 running in the unconfined\_t domain. Eventually we hope to get most of the daemons   
 in question working well in more restrictive domains. The main benefit, however, in  
 terms of the usability of the targeted policy is in the lack of restrictive domains  
 for user sessions. It seems likely that the demand for this feature will exist for   
 a long time. Therefore merging the strict and targeted policies will not be   
 possible.  
 Supported configuration changes  
   
  
Within the targeted policy, significant changes will break the support agreement for   
 Red Hat Enterprise Linux. This means modifying the policy for daemons (particularly  
 if such changes involve reducing access) and adding new domains for programs that   
 are part of Red Hat Enterprise Linux. Adding new changes for programs that are not   
 part of Red Hat Enterprise Linux may be OK as long as any bugs which are reported   
 do not concern the programs in question.  
   
  
If you go outside the supported configuration either through excessive changes to the  
 targeted policy or through using the strict policy, support for SELinux features   
 will be provided through GPS (Red Hat consulting). For support on issues unrelated   
 to SELinux you may have to put SELinux in permissive mode when reporting problems.  
 Strict policy support  
   
  
The Red Hat Enterprise Linux 4 release only contains the policy packages for the   
 targeted policy because it is the only policy supported through the Global Support   
 Services.  
   
  
Packages for the strict policy will be available on the Red Hat website for   
 organizations that have a policy of having all their system software come from the   
 one source. They can get the strict policy from Red Hat, it just won't be supported  
 through the usual support processes Anyone who wants to run SELinux on a Red Hat   
 Enterprise Linux 4 system will be able to download the package   
 selinux-policy-strict (and selinux-policy-strict-sources if they want to modify the  
 policy source) and convert their system to the strict policy.  
   
  
To convert a Fedora or Red Hat Enterprise Linux 4 system to the strict policy you   
 only have to install the strict policy package and run system-config-securitylevel   
 in an X session. You will then see a tab to configure SELinux. The SELinux tab   
 features a drop-down list box that allows you to select between the installed   
 policies as shown in Figure 2, Select between installed policies.  
   
  
After selecting this option, you should reboot the machine at the earliest convenient  
 time. Early in the boot process the script /etc/rc.sysinit will relabel the file   
 system with the correct labels for the new policy type. Currently the configuration  
 file /etc/selinux/config has one field for which type of policy is to be used,   
 indicating which will be used on the next boot. The same field is used by some   
 applications to determine which policy is currently running. So in the period   
 between changing the policy through system-config-securitylevel and rebooting to   
 apply the change, some programs may not operate in the desired manner as their idea  
 of the running policy will not match reality. This is not a security issue as they   
 will fail closed, but it may be a usability issue. One consequence of having the   
 running policy not match the policy that is configured for the next boot is that   
 cron jobs will not run.  
   
  
The process of relabeling the file system involves comparing the fully qualified path  
 name of each file on the system with a set of regular expressions such that the   
 best match will indicate which security context should be assigned to the file.   
 Thus the file system relabel process involved with converting between strict and   
 targeted policies. This process will take at least as much time as find / and maybe  
 as much as twice that due to the amount of computation taken for regular   
 expressions. With a typical Red Hat Enterprise Linux or Fedora install on modern   
 hardware this should only take a few minutes. If you have many of your own files   
 installed then it will take proportionally longer.  
   
  
If you have a server with millions of files with the same security context on their   
 own file system, then it's best to use the context mount option to label them. This  
 saves the time taken for a relabel and also saves the storage requirements for the   
 security labels. For example the Squid cache files are labeled as   
 system\_u:object\_r:squid\_cache\_t. If you had a large Squid server with a file system  
 devoted to Squid, then you could put fscontext=system\_u:object\_r:squid\_cache\_t in   
 the file system options field in the /etc/fstab file.  
   
  
Red Hat Enterprise Linux 4 systems which use the strict policy will only be supported  
 for non-SE functionality. Customers who make such modifications to their systems   
 may be requested to put SELinux in permissive mode and reproduce the problem when   
 making a support call for issues that are not directly related to SELinux.   
 Permissive mode is when SELinux reports that it would not permit an operation but   
 does not actually prevent the operation from occurring. Permissive mode is used for  
 development of SELinux policies and for many types of testing. To quickly determine  
 whether SELinux is the cause of a problem it can be put in permissive mode by the   
 command setenforce 0 and then put back in enforcing mode with setenforce 1 after   
 the test is complete. Note that it is not recommended that you put a production   
 machine into permissive mode.  
   
  
At this time support for the strict policy is only provided through GPS (the Red Hat   
 consulting division). The regular support channels will not accept calls about it,   
 and the guarantees about response time also do not apply. The strict policy is not   
 on the Red Hat Enterprise Linux 4 CDs and is officially not part of the   
 distribution.  
   
The /etc/selinux/ directory  
   
  
In Fedora Core 3 the SELinux base directory was changed to /etc/selinux/, and this is  
 the location that is used for Red Hat Enterprise Linux 4 and will be used for all   
 future releases. In the directory /etc/selinux/targeted/ you will find the files   
 for the targeted policy. If you are running the strict policy, you will have the   
 /etc/selinux/strict/ directory. By default, the policy sources are not installed.   
 To install the policy sources, you need the selinux-policy-targeted-sources package  
 (if running the strict policy, you - need the selinux-policy-strict-sources   
 package). The installation of this package results in the   
 /etc/selinux/targeted/src/ (or strict/src/) directory being installed. Here (under   
 the policy directory) you will find the policy source.  
 User roles in the targeted policy  
   
  
Under the targeted policy, there is no real use of user roles and domains. All user   
 identities are permitted to use the role system\_r, and thus identities and roles   
 play no part in SELinux access control. Under the strict policy, every user who is   
 significant to the system security policy (one category of which is users who are   
 granted administrative rights) needs an entry in the users file to specify the   
 roles which they are permitted to assume. With the targeted policy this aspect of   
 policy configuration is not needed.  
 Development of other policies  
   
  
The design of SELinux has all configuration options in the SELinux policy, and no   
 configuration options are compiled into binaries. This means that the complete   
 configuration of SELinux can be changed without changing any programs.   
 Administrators of Red Hat Enterprise Linux 4 systems are free to build their own   
 policy that is not based on either the strict or targeted policies. But again, this  
 would be outside the Red Hat Enterprise Linux 4 support contracts.

======================================  
 SUDO:  
======================================  
  
 Before we proceed, it would be best to cover some basic user administration topics   
 that will be very useful in later chapters. Adding Users   
   
One of the most important activities in administering a Linux box is the addition of   
 users. Here you'll find some simple examples to provide a foundation for future   
 chapters. It is not intended to be comprehensive, but is a good memory refresher.   
 You can use the command man useradd to get the help pages on adding users with the   
 useradd command or the man usermod to become more familiar with modifying users   
 with the usermod command.  
   
Who Is the Super User?   
   
The super user with unrestricted access to all system resources and files in Linux is  
 the user named root. This user has a user ID, of 0 which is universally identified   
 by Linux applications as belonging to a user with supreme privileges. You will need  
 to log in as user root to add new users to your Linux server.   
   
Debian Note: When installing Ubuntu Linux systems, you are prompted to create a   
 primary user that is not root. A root user is created but no password is set, so   
 you initially cannot log in as this user. The primary user can become the root user  
 using the sudo su - command that will be discussed later.  
   
How To Add Users   
   
Adding users takes some planning; read through these steps below before starting:   
   
1) Arrange your list of users into groups by function. In this example there are   
 three groups "parents", "children" and "soho".   
   
Parents Children Soho   
   
Paul Alice Accounts   
Jane Derek Sales   
   
2) Add the Linux groups to your server:   
   
[root@bigboy tmp]# groupadd parents   
[root@bigboy tmp]# groupadd children   
[root@bigboy tmp]# groupadd soho   
   
3) Add the Linux users and assign them to their respective groups   
   
[root@bigboy tmp]# useradd -g parents paul   
[root@bigboy tmp]# useradd -g parents jane   
[root@bigboy tmp]# useradd -g children derek   
[root@bigboy tmp]# useradd -g children alice   
[root@bigboy tmp]# useradd -g soho accounts   
[root@bigboy tmp]# useradd -g soho sales   
   
If you don't specify the group with the -g, RedHat/Fedora Linux creates a group with   
 the same name as the user you just created; this is also known as the User Private   
 Group Scheme. When each new user first logs in, they are prompted for their new   
 permanent password.   
   
4) Each user's personal directory is placed in the /home directory. The directory   
 name will be the same as their user name.   
   
[root@bigboy tmp]# ll /home   
drwxr-xr-x 2 root root 12288 Jul 24 20:04 lost+found   
drwx------ 2 accounts soho 1024 Jul 24 20:33 accounts   
drwx------ 2 alice children 1024 Jul 24 20:33 alice   
drwx------ 2 derek children 1024 Jul 24 20:33 derek   
drwx------ 2 jane parents 1024 Jul 24 20:33 jane   
drwx------ 2 paul parents 1024 Jul 24 20:33 paul   
drwx------ 2 sales soho 1024 Jul 24 20:33 sales   
[root@bigboy tmp]#   
   
   
How to Change Passwords   
   
You need to create passwords for each account. This is done with the passwd command.   
 You are prompted once for your old password and twice for the new one.   
   
 \* User root changing the password for user paul.   
   
[root@bigboy root]# passwd paul   
Changing password for user paul.   
New password:   
Retype new password:   
passwd: all authentication tokens updated successfully.   
[root@bigboy root]#   
   
 \* Users might wish to change their passwords at a future date. Here is how   
 unprivileged user paul would change his own password.   
   
[paul@bigboy paul]$ passwd   
Changing password for paul   
Old password: your current password   
Enter the new password (minimum of 5, maximum of 8 characters)   
Please use a combination of upper and lower case letters and numbers.   
New password: your new password   
Re-enter new password: your new password   
Password changed.   
[paul@bigboy paul]$   
   
   
How to Delete Users   
   
The userdel command is used to remove the user's record from the /etc/passwd and   
 /etc/shadow used in the login process. The command has a single argument, the   
 username.   
   
[root@bigboy tmp]# userdel paul   
   
There is also an optional -r switch that additionally removes all the contents of the  
 user's home directory. Use this option with care. The data in a user's directory   
 can often be important even after the person has left your company.   
   
[root@bigboy tmp]# userdel -r paul   
   
   
How to Tell the Groups to Which a User Belongs   
   
Use the groups command with the username as the argument.   
   
[root@bigboy root]# groups paul   
paul : parents   
[root@bigboy root]#   
   
   
How to Change the Ownership of a File   
   
You can change the ownership of a file with the chown command. The first argument is   
 the desired username and group ownership for the file separated by a colon (:)   
 followed by the filename. In the next example we change the ownership of the file   
 named text.txt from being owned by user root and group root to being owned by user   
 testuser in the group users:   
   
[root@bigboy tmp]# ll test.txt   
-rw-r--r-- 1 root root 0 Nov 17 22:14 test.txt   
[root@bigboy tmp]# chown testuser:users test.txt   
[root@bigboy tmp]# ll test.txt   
-rw-r--r-- 1 testuser users 0 Nov 17 22:14 test.txt   
[root@bigboy tmp]#   
   
You can also use the chown command with the -r switch for it to doe recursive   
 searches down into directories to change permissions.   
   
   
Using sudo   
   
If a server needs to be administered by a number of people it is normally not a good   
 idea for them all to use the root account. This is because it becomes difficult to   
 determine exactly who did what, when and where if everyone logs in with the same   
 credentials. The sudo utility was designed to overcome this difficulty.   
   
The sudo utility allows users defined in the /etc/sudoers configuration file to have   
 temporary access to run commands they would not normally be able to due to file   
 permission restrictions. The commands can be run as user "root" or as any other   
 user defined in the /etc/sudoers configuration file.   
   
The privileged command you want to run must first begin with the word sudo followed   
 by the command's regular syntax. When running the command with the sudo prefix, you  
 will be prompted for your regular password before it is executed. You may run other  
 privileged commands using sudo within a five-minute period without being   
 re-prompted for a password. All commands run as sudo are logged in the log file   
 /var/log/messages.   
   
   
Simple Sudo Examples   
   
Using sudo is relatively simple as we can see from these examples.   
Temporarily Gaining root Privileges   
   
In this example, user bob attempts to view the contents of the /etc/sudoers file,   
 which is an action that normally requires privileged access. Without sudo, the   
 command fails:   
   
[bob@bigboy bob]$ more /etc/sudoers   
/etc/sudoers: Permission denied   
[bob@bigboy bob]$   
   
Bob tries again using sudo and his regular user password and is successful:   
   
[bob@bigboy bob]$ sudo more /etc/sudoers   
Password:   
...   
...   
[bob@bigboy bob]$   
   
The details of configuring and installing sudo are covered in later sections.   
Becoming root for a Complete Login Session   
   
The su command allows a regular user to become the system's root user if they know   
 the root password. A user with sudo rights to use the su command can become root,   
 but they only need to know their own password, not that of root as seen here.   
   
someuser@u-bigboy:~$ sudo su -   
Password:   
root@u-bigboy:~#   
   
Some systems administrators will use sudo to grant root privileges to their own   
 personal user account without the need to provide a password.   
   
Later sections describe how to disable sudo su ability and also how to use sudo   
 without password prompts.   
Downloading and Installing the sudo Package   
   
Fortunately the package is installed by default by RedHat/Fedora which eliminates the  
 need to anything more in this regard. The visudo Command   
   
The visudo command is a text editor that mimics the vi editor that is used to edit   
 the /etc/sudoers configuration file. It is not recommended that you use any other   
 editor to modify your sudo parameters because the sudoers file isn't located in the  
 same directory on all versions of Linux. visudo uses the same commands as the vi   
 text editor. The visudo command must run as user root and should have no   
 arguments:   
   
[root@aqua tmp]# visudo   
   
   
The /etc/sudoers File   
   
The /etc/sudoers file contains all the configuration and permission parameters needed  
 for sudo to work. There are a number of guidelines that need to be followed when   
 editing it with visudo. General /etc/sudoers Guidelines   
   
The /etc/sudoers file has the general format shown in Table 9-1.   
   
   
Table 9-1 Format of the /etc/sudoers File   
   
General sudoers File Record Format   
   
usernames/group servername = (usernames command can be run as) command   
   
There are some general guidelines when editing this file:   
   
 \* Groups are the same as user groups and are differentiated from regular users by  
 a % at the beginning. The Linux user group "users" would be represented by   
 %users.   
 \* You can have multiple usernames per line separated by commas.   
 \* Multiple commands also can be separated by commas. Spaces are considered part   
 of the command.   
 \* The keyword ALL can mean all usernames, groups, commands and servers.   
 \* If you run out of space on a line, you can end it with a back slash (\) and   
 continue on the next line.   
 \* sudo assumes that the sudoers file will be used network wide, and therefore   
 offers the option to specify the names of servers which will be using it in the  
 servername position in Table 9-1. In most cases, the file is used by only one   
 server and the keyword ALL suffices for the server name.   
 \* The NOPASSWD keyword provides access without prompting for your password.   
   
   
Simple /etc/sudoers Examples   
   
This section presents some simple examples of how to do many commonly required tasks   
 using the sudo utility.  
   
Granting All Access to Specific Users   
   
You can grant users bob and bunny full access to all privileged commands, with this   
 sudoers entry.   
   
bob, bunny ALL=(ALL) ALL   
   
This is generally not a good idea because this allows bob and bunny to use the su   
 command to grant themselves permanent root privileges thereby bypassing the command  
 logging features of sudo. The example on using aliases in the sudoers file shows   
 how to eliminate this problem  
   
Granting Access To Specific Users To Specific Files   
   
This entry allows user peter and all the members of the group operator to gain access  
 to all the program files in the /sbin and /usr/sbin directories, plus the privilege  
 of running the command /usr/local/apps/check.pl. Notice how the trailing slash (/)   
 is required to specify a directory location:   
   
peter, %operator ALL= /sbin/, /usr/sbin, /usr/local/apps/check.pl   
   
Notice also that the lack of any username entries within parentheses () after the =   
 sign prevents the users from running the commands automatically masquerading as   
 another user. This is explained further in the next example.   
   
Granting Access to Specific Files as Another User   
   
The sudo -u entry allows allows you to execute a command as if you were another user,  
 but first you have to be granted this privilege in the sudoers file.   
   
This feature can be convenient for programmers who sometimes need to kill processes   
 related to projects they are working on. For example, programmer peter is on the   
 team developing a financial package that runs a program called monthend as user   
 accounts. From time to time the application fails, requiring "peter" to stop it   
 with the /bin/kill, /usr/bin/kill or /usr/bin/pkill commands but only as user   
 "accounts". The sudoers entry would look like this:   
   
peter ALL=(accounts) /bin/kill, /usr/bin/kill, /usr/bin/pkill   
   
User peter is allowed to stop the monthend process with this command:   
   
[peter@bigboy peter]# sudo -u accounts pkill monthend   
   
Granting Access Without Needing Passwords   
   
This example allows all users in the group operator to execute all the commands in   
 the /sbin directory without the need for entering a password. This has the added   
 advantage of being more convenient to the user:   
   
%operator ALL= NOPASSWD: /sbin/   
   
Using Aliases in the sudoers File   
   
Sometimes you'll need to assign random groupings of users from various departments   
 very similar sets of privileges. The sudoers file allows users to be grouped   
 according to function with the group and then being assigned a nickname or alias   
 which is used throughout the rest of the file. Groupings of commands can also be   
 assigned aliases too.   
   
In the next example, users peter, bob and bunny and all the users in the operator   
 group are made part of the user alias ADMINS. All the command shell programs are   
 then assigned to the command alias SHELLS. Users ADMINS are then denied the option   
 of running any SHELLS commands and su:   
   
Cmnd\_Alias SHELLS = /usr/bin/sh, /usr/bin/csh, \   
 /usr/bin/ksh, /usr/local/bin/tcsh, \   
 /usr/bin/rsh, /usr/local/bin/zsh   
   
   
User\_Alias ADMINS = peter, bob, bunny, %operator   
ADMINS ALL = !/usr/bin/su, !SHELLS   
   
This attempts to ensure that users don't permanently su to become root, or enter   
 command shells that bypass sudo's command logging. It doesn't prevent them from   
 copying the files to other locations to be run. The advantage of this is that it   
 helps to create an audit trail, but the restrictions can be enforced only as part   
 of the company's overall security policy.   
Other Examples   
   
You can view a comprehensive list of /etc/sudoers file options by issuing the command  
 man sudoers.   
Using syslog To Track All sudo Commands   
   
All sudo commands are logged in the log file /var/log/messages which can be very   
 helpful in determining how user error may have contributed to a problem. All the   
 sudo log entries have the word sudo in them, so you can easily get a thread of   
 commands used by using the grep command to selectively filter the output   
 accordingly.   
   
Here is sample output from a user bob failing to enter their correct sudo password   
 when issuing a command, immediately followed by the successful execution of the   
 command /bin/more sudoers.   
   
[root@bigboy tmp]# grep sudo /var/log/messages   
 Nov 18 22:50:30 bigboy sudo(pam\_unix)[26812]: authentication failure;   
 logname=bob uid=0 euid=0 tty=pts/0 ruser= rhost= user=bob   
 Nov 18 22:51:25 bigboy sudo: bob : TTY=pts/0 ; PWD=/etc ;   
 USER=root ; COMMAND=/bin/more sudoers   
 [root@bigboy tmp]#   
   
   
Conclusion   
   
It is important to know how to add users, not just so they can log in to our system.   
 Most server based applications usually run via a dedicated unprivileged user   
 account, for example the MySQL database application runs as user mysql and the   
 Apache Web server application runs as user apache. These accounts aren't always   
 created automatically, especially if the software is installed using TAR files.   
   
Finally, the sudo utility provides a means of dispersing the responsibility of   
 systems management to multiple users. You can even give some groups of users only   
 partial access to privileged commands depending on their roles in the organization.  
 This makes sudo a valuable part of any company's server administration and security  
 policy.  
  
======================================   
SSH:   
======================================   
   
SSH uses public-key cryptography to authenticate the remote computer and allow the   
 remote computer to authenticate the user, if necessary.[1]   
   
SSH is typically used to log into a remote machine and execute commands, but it also   
 supports tunneling, forwarding TCP ports and X11 connections; it can transfer files  
 using the associated SFTP or SCP protocols.[1] SSH uses the client-server model.   
   
An SSH server, by default, listens on the standard TCP port 22.[3]   
   
An SSH client program is typically used for establishing connections to an SSH daemon  
 accepting remote connections. Both are commonly present on most modern operating   
 systems, including Mac OS X, Linux, FreeBSD, Solaris and OpenVMS. Proprietary,   
 freeware and open source versions of various levels of complexity and completeness   
 exist.   
   
Generate Trusted Keys:   
   
ssh-keygen -t dsa   
Generating public/private dsa key pair.   
Enter file in which to save the key (/home/localuser/.ssh/id\_dsa):   
Enter passphrase (empty for no passphrase):   
Enter same passphrase again:   
Your identification has been saved in /home/localuser/.ssh/id\_dsa.   
Your public key has been saved in /home/localuser/.ssh/id\_dsa.pub.   
The key fingerprint is: 93:58:20:56:72:d7:bd:14:86:9f:42:aa:82:3d:f8:e5 localuser   
Administrator@helios ~/.ssh scp ~/.ssh/id\_dsa.pub root@sirius:.ssh/authorized\_keys   
The authenticity of host 'sirius (192.168.1.4)' can't be established.   
RSA key fingerprint is 39:ba:5c:9d:a9:05:fa:f2:ba:67:bd:d5:9e:f1:95:8c.   
Are you sure you want to continue connecting (yes/no)? yes   
Warning: Permanently added 'sirius' (RSA) to the list of known hosts.   
Warning: the RSA host key for 'sirius' differs from the key for the IP address   
 '192.168.1.4'   
Offending key for IP in /home/Administrator/.ssh/known\_hosts:1   
Are you sure you want to continue connecting (yes/no)? yes   
Password:   
id\_dsa.pub   
   
Copy File From Server:   
Administrator@helios ~   
$ scp root@vega:install.log .   
install.log 100% 25KB 25.4KB/s 00:00   
   
   
   
Copy File to Server:   
   
Administrator@helios ~   
$ scp ./rotatelog.zip root@vega:   
rotatelog.zip 100% 6658 6.5KB/s 00:00   
   
NOTE: Notice I didn't need to authenticate since I had already copied the trusted   
 keys I generated over to root on Vega's ~/.ssh directory.   
   
   
   
   
======================================   
HARDENING:   
======================================   
   
   
   
   
======================================   
FIREWALL:   
======================================   
   
  
  
  
======================================  
 IPCHAINS / IPTABLES:  
======================================  
   
Firewall versions vs Linux versions:  
   
Note: References to ipfwadm and ipchains refer to older deprecated software.  
   
Command  
 =======  
 iptables  
 ipchains  
 ipfwadm  
   
Kernel Ver.  
 ===========  
 2.4.x, 2.6.x  
 2.2.x  
 2.0.x  
   
Red Hat Version  
 ===============  
 7.1 - 9.0, Fedora 1,2,3  
 6.x, 7.0  
 5.x  
   
Note: Red Hat 7.1-9.0 and the default Linux 2.4 kernel may use ipchains or iptables   
 but not both. Iptables is the preferred firewall as it supports "state" and can   
 recognize if a network connection has already been "ESTABLISHED" or if the   
 connection is related to the previous connection (required for ftp which makes   
 multiple connections on different ports). Ipchains can not. Ipchain rules take   
 precedence over iptables rules. During system boot, the kernel attempts to activate  
 ipchains, then attempts to activate iptables. If ipchain rules have been activated,  
 the kernel will not start iptables.  
   
Red Hat 7.1 will not support ipchains unless that option is configured (during   
 install or later). If during install you select "Disable Firewall - no protection"   
 then ipchains will not be available and you must rely upon iptables for a manual   
 firewall configuration. (iptables only. ipchains will be unavailable)  
   
GUI configuration:  
   
\* iptables: The GUI configuration tool /usr/bin/redhat-config-securitylevel can be   
 used to choose a preconfigured firewall (High, Medium or no firewall) or it can be   
 used to manually configure rules based on the network services your server will   
 offer. The init script /etc/rc.d/init.d/iptables will use rules stored in   
 /etc/sysconfig/iptables.  
  
 \* ipchains: The tool that does this is lokkit (or /usr/bin/gnome-lokkit), which uses   
 ipchains to configure firewall options for High and Low security options. To   
 support ipchains after install, run /usr/bin/gnome-lokkit and configure a firewall.  
 It will configure ipchains to activate the firewall. Lokkit will generate the file   
 /etc/sysconfig/ipchains. (Used by init script /etc/rc.d/init.d/ipchains which calls  
 /sbin/ipchains-restore)  
   
To see if ipchains and the Lokkit configuration is invoked during system boot, use   
 the command:  
   
chkconfig --list | grep ipchains  
   
The default Red Hat 7.1+ Linux 2.4 kernel is compiled to support both iptables and   
 ipchains. Kernel support for ipchains is available during a kernel configuration   
 and compilation. During make xconfig or make menuconfig turn on the feature: "IP:   
 Netfilter Configuration" + "ipchains (2.2-style) support".  
   
Check your installation by using the command: rpm -q iptables ipchains  
 These packages must be installed. The commands iptables and ipchains are the command   
 interfaces to configure kernel firewall rules. The default Red Hat 7.1 kernel   
 supports iptables and ipchains. (But not both at the same time.)  
   
[Potential Pitfall]: When performing an upgrade instead of a new install, the upgrade  
 software will not install iptables as did not exist on the system previously. It   
 will perform an upgrade to a newer version of ipchains. If you wish to use   
 iptables, you must manually install the iptables RPM.  
 i.e.: rpm -ivh iptables-XXX.i386.rpm  
   
[Potential Pitfall]: The Linux operating system kernel may load or not load what you   
 had expected. Use the command lsmod to see if ip\_tables or ip\_chains were loaded.  
   
Switching a running system from ipchains to iptables: (Red Hat 7.1-9.0 - Linux kernel  
 2.4 specific)  
   
Sequence 1  
 Command chkconfig --del ipchains  
 Description Remove ipchains from system boot/initialization process  
   
Sequence 2  
 Command chkconfig --add iptables  
 Description Add iptables to system boot/initialization process  
   
Sequence 3  
 Command ipchains -F  
 Description Flush ipchains rules  
   
Sequence 4  
 Command service ipchains stop  
 Description Stop ipchains. Also: /etc/init.d/ipchains stop  
   
Sequence 5  
 Command rmmod ipchains  
 Description Unload ipchains kernel module. Iptables kernel module can not be loaded   
 if the ipchains module is loaded.  
   
Sequence 6  
 Command service iptables start  
 Description Load iptables kernel module. Also: /etc/init.d/iptables stop  
  
======================================  
 STORAGE///////////////////////////  
======================================  
  
====================================  
 SOLARIS DISK COMMANDS:  
====================================  
   
/bin/mount -F hsfs -o ro /dev/sr0 /cdrom  
   
/\* Mount an ISO 9660 CDROM \*/  
=======================================  
   
/usr/bin/iostat -E  
   
/\* Command to display drives statistics \*/  
=======================================  
   
du -ad /var | sort -nr  
   
/\* Report the the disk used in /var in reverse order \*/  
=======================================  
   
du -k .  
   
/\* Report disk usage in Kilobytes \*/  
=======================================  
   
du -sk \* | sort -nr | head  
   
/\* Shows the top ten largest files/directories \*/  
=======================================  
   
du -sk \*|sort -k1,1n  
   
/\* Reports total disk space used in Kilobytes in present directory \*/  
=======================================  
   
du -sk .  
   
/\* Report total disk usage in Kilobytes \*/  
=======================================  
   
fdformat -d -U  
   
/\* Format diskette \*/  
=======================================  
   
mount -F hsfs -o ro `lofiadm -a /export/temp/software.iso` /mnt  
   
/\* Mount an ISO Image \*/  
=======================================  
   
newfs -Nv /dev/rdsk/c0t0d0s1  
   
/\* To view the superfblocks available \*/  
=======================================  
   
One-liner to copy a partition table  
   
/\* prtvtoc /dev/dsk/c1t2d0s2 | fmthard -s - /dev/rdsk/c1t3d0s2 \*/  
=======================================  
   
prtvtoc /dev/rdsk/c0t0d0s2  
   
/\* Disk geometry and partitioning info \*/  
=======================================  
   
prtvtoc /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c0t1d0s2  
   
/\* Copy partition table from one disk to another \*/  
=======================================  
   
quot -af  
   
/\* How much space is used by users in kilobytes \*/  
=======================================  
   
volrmmount -i floppy  
   
/\* Mount a floppy or other media easily by its nickname. \*/  
=======================================  
   
Driver Parameters  
   
ndd /dev/ip ip\_forwarding  
   
/\* Show the ip\_forwarding variable in the kernel \*/  
=======================================  
   
ndd /dev/ip ip\_forwarding 1  
   
/\* Set the ip\_forwarding variable in the kernel \*/  
=======================================  
   
ndd /dev/ip \?  
   
/\* Show all IP variables set in the kernel \*/  
======================================  
  
  
======================================  
 LUN:  
======================================  
   
(Logical Unit Number) An identification scheme for storage disks that typically   
 supports a small number of units addressed as LUN 0 through 7, 15 or 31 depending   
 on the technology. For example, Fibre Channel supports 32 addresses (0-31). A LUN   
 may refer to a single disk, a subset of a single disk or an array of disks. Derived  
 from the SCSI bus technology, each SCSI ID address can be further subdivided into   
 LUNs 0 through 15 for disk arrays and libraries. See SCSI.  
  
======================================  
 iSCSI  
======================================  
   
By carrying SCSI commands over IP networks, iSCSI is used to facilitate data   
 transfers over intranets and to manage storage over long distances. The iSCSI   
 protocol is among the key technologies expected to help bring about rapid   
 development of the storage area network (SAN) market, by increasing the   
 capabilities and performance of storage data transmission. Because of the ubiquity   
 of IP networks, iSCSI can be used to transmit data over local area networks (LANs),  
 wide area networks (WANs), or the Internet and can enable location-independent data  
 storage and retrieval.  
  
======================================  
 SAN  
======================================  
 How do you make a LUN visible to a host on a SAN?~  
======================================  
 SAN Solaris:  
======================================  
   
Ensuring That LUN Level Information Is Visible  
 To Ensure LUN Level Information is Visible  
   
Use the cfgadm command to identify LUN level information.  
   
If you issue the cfgadm -al -o show\_SCSI\_LUN controller-ID command immediately after   
 a system boots up, the output might not show the Fibre Channel Protocol (FCP) SCSI   
 LUN level information. The information does not appear because the storage device   
 drivers, such as the ssd and st driver, are not loaded on the running system.  
   
Use the modinfo command to check whether the drivers are loaded. After the drivers   
 are loaded, the LUN level information is visible in the cfgadm output.  
 To Detect Fabric Devices Visible on a Host  
   
This section provides an example of the procedure for detecting fabric devices using   
 FC host ports c0 and c1. This procedure also shows the device configuration   
 information that is displayed with the cfgadm(1M) command.  
 Note:  
   
In the following examples, only failover path attachment point IDs (Ap\_Ids) are   
 listed. The Ap\_Ids displayed on your system depend on your system configuration.  
   
Log in as root (su - root)..  
   
Display the information about the attachment points on the system.  
 # cfgadm -l  
 Ap\_Id Type Receptacle Occupant Condition  
 c0 fc-fabric connected unconfigured unknown  
 c1 fc-private connected configured unknown  
   
In this example, c0 represents a fabric-connected host port, and c1 represents a   
 private, loop-connected host port. Use the cfgadm(1M) command to manage the device   
 configuration on fabric-connected host ports.  
   
By default, the device configuration on private, loop-connected host ports is managed  
 by a host using the Solaris Express Developer's Edition OS.  
   
Display information about the host ports and their attached devices.  
 # cfgadm -al  
 Ap\_Id Type Receptacle Occupant Condition  
 c0 fc-fabric connected unconfigured unknown  
 c0::50020f2300006077 disk connected unconfigured unknown  
 c0::50020f23000063a9 disk connected unconfigured unknown  
 c0::50020f2300005f24 disk connected unconfigured unknown  
 c0::50020f2300006107 disk connected unconfigured unknown  
 c1 fc-private connected configured unknown  
 c1::220203708b69c32b disk connected configured unknown  
 c1::220203708ba7d832 disk connected configured unknown  
 c1::220203708b8d45f2 disk connected configured unknown  
 c1::220203708b9b20b2 disk connected configured unknown  
   
Note:  
   
The cfgadm -l command displays information about FC host ports. You can also use the   
 cfgadm -al command to display information about FC devices. The lines that include   
 a port world wide name (WWN) in the Ap\_Id field associated with c0 represent a   
 fabric device. Use the cfgadm configure and unconfigure commands to manage those   
 devices and make them available to hosts using the Solaris Express Developer's   
 Edition OS. The Ap\_Id devices with port WWNs under c1 represent private-loop   
 devices that are configured through the c1 host port.  
  
======================================  
 SAN Linux qla-scan:  
======================================  
   
1) First the devices (LUN's) are to be detected by HBA's  
 We do this by scanning BOTH the HBA's using qla-scan?  
   
2) We can find the new LUNs in /proc/scsi/qla200/ 0  
 /proc/scsi/qla200/ 1  
   
(Id:Lun) \* - indicates lun is not registered with the OS.  
   
( 0: 0): Total reqs 1, Pending reqs 0, flags 0x0\*, 0:0:81,  
   
( 0:15): Total reqs 0, Pending reqs 0, flags 0x0\*, 0:0:81,  
   
If a new LUN is added, after a scan i may see something like  
   
(0:16): Total reqs 0, Pending reqs 0, flags 0x0\*, 0:0:81, where 0 is the  
 target ID and 16 is LUN ???????  
   
(( echo 1 > /sys/class/fc\_ host/hostn/ issue\_lip  
 echo "- - -" > /sys/class/scsi\_ host/host2/ scan  
   
works for upstream kernels )  
   
3) echo scsi add-single-device 2 0 0 16 > /proc/scsi/scsi so the new LUN  
 discovered at HBA 2, BUS 0, TARGET 0, LUN ID 16 will be understood by the  
 kernel/OS ?#  
   
4) Now we do a normal fdisk...  
  
======================================  
 SAN Linux Q&A:  
======================================  
   
How is HBA and LUN order persistent? Where is this information stored?  
   
1) In Linux mostly HBA and LUN order is Persistant ( by default), it will be stored   
 in /proc/scsi/<Driver Name > and /etc/modules.conf  
   
How are the naming conventions done on the newly added scsi LUN?  
   
2) Assume my first INTERNAL DISK was sda.  
 - The Newly added device might be appear to you as sdb  
   
How will the new LUN's be named?  
   
3) A LUN is basically named or idendified by WWW number( 16 digit ) which will   
 communicate with SAN switch and Disk. Make sure that newly configured LUN's are   
 properly configured and mapped to correct HBA's  
  
======================================  
 SAN Linux How To:  
======================================  
   
1. If you have installed SANsurfer GUI/CLI, make sure newly added LUN's are visible   
 from these tools.  
   
2. For each QLogic HBA installed in the system, do the following:  
   
a. Determine the host ID (H) associated with the HBA. Each installed HBA has a  
   
numeric filename that is the host identifier.  
   
# /bin/ls /proc/scsi/qla2300  
   
b. Rescan for all the LUNs on all the ports.  
   
H is the host identifier associated with HBA.  
   
# /bin/echo scsi-qlascan > /proc/scsi/qla2300/ H  
   
c. Determine the target ID (T) associated with the new LUN.  
   
This file lists the ID:LUN numbers recognized by the QLA driver. T is the  
   
target ID value.  
   
# /bin/cat /proc/scsi/qla2300/ H  
   
d. Add the LUN to HBA.  
   
H is the host identifier associated with the HBA;  
   
T is the target identifier obtained in Step c; and L is the LUN identifier.  
   
# /bin/echo scsi add-single-device H 0 T L > /proc/scsi/scsi  
   
4. use "/sbin/fdisk -l" to verify if the newly added LUN is visible to the OS  
   
5. use InfoDoc 85804 "How to create a new partition table under Redhat Linux"  
   
for creating properly a partition and a file system on the LUN  
  
======================================  
 SAN Linux HBA Example:  
======================================  
   
Find below example illustrating addition of a newly mapped T4 lun on RedHat  
   
Linux using QLA driver version 7.07.04  
   
1) Determine host ID  
   
root]# /bin/ls /proc/scsi/qla2300/  
   
2  
   
2) Re-scan for LUN's on both HBA's  
   
root]# /bin/echo scsi-qlascan > /proc/scsi/qla2300/ 2  
   
3) Verify newly mapped LUN's  
   
root]# /bin/cat /proc/scsi/qla2300/ 2  
   
QLogic PCI to Fibre Channel Host Adapter for QLA2342:  
   
Firmware version: 3.03.19, Driver version 7.07.04  
   
Entry address = f88ae060  
   
HBA: QLA2312 , Serial# F05179  
   
Request Queue = 0x377e0000, Response Queue = 0x377d0000  
   
Request Queue count= 512, Response Queue count= 512  
   
Total number of active commands = 0  
   
Total number of interrupts = 35  
   
Total number of active IP commands = 0  
   
Total number of IOCBs (used/max) = (0/600)  
   
Total number of queued commands = 0  
   
Device queue depth = 0x20  
   
Number of free request entries = 510  
   
Number of mailbox timeouts = 0  
   
Number of ISP aborts = 0  
   
Number of loop resyncs = 2  
   
Number of retries for empty slots = 0  
   
Number of reqs in pending\_q= 0, retry\_q= 0, done\_q= 0, scsi\_retry\_q= 0  
   
Host adapter:loop state= <READY>, flags= 0x820813  
   
Dpc flags = 0x0  
   
MBX flags = 0x0  
   
SRB Free Count = 4096  
   
Link down Timeout = 000  
   
Port down retry = 030  
   
Login retry count = 030  
   
Commands retried with dropped frame(s) = 0  
   
Configured characteristic impedence: 50 ohms  
   
Configured data rate: 1-2 Gb/sec auto-negotiate  
   
SCSI Device Information:  
   
scsi-qla0-adapter- node=200100e08b2 75bb5;  
   
scsi-qla0-adapter- port=210100e08b2 75bb5;  
   
scsi-qla0-target- 0=20030003ba27cf a2;  
   
SCSI LUN Information:  
   
(Id:Lun) \* - indicates lun is not registered with the OS.  
   
( 0: 0): Total reqs 1, Pending reqs 0, flags 0x0\*, 0:0:81,  
   
( 0:15): Total reqs 0, Pending reqs 0, flags 0x0\*, 0:0:81,  
   
Note "\*" in the above outputs, which indicates lun as not registered  
   
with OS, hence not visible to OS.  
   
4) Register the new LUN's with OS  
   
# /bin/echo scsi add-single-device 2 0 0 15 > /proc/scsi/scsi  
   
5) verify from OS using "fdisk -l"  
   
root]# /sbin/fdisk -l  
   
Disk /dev/sda: 36.4 GB, 36420075008 bytes  
   
255 heads, 63 sectors/track, 4427 cylinders  
   
Units = cylinders of 16065 \* 512 = 8225280 bytes  
   
Device Boot Start End Blocks Id System  
   
/dev/sda1 \* 1 13 104391 83 Linux  
   
/dev/sda2 14 4173 33415200 83 Linux  
   
/dev/sda3 4174 4427 2040255 82 Linux swap  
   
Disk /dev/sdb: 5372 MB, 5372116992 bytes  
   
255 heads, 63 sectors/track, 653 cylinders  
   
Units = cylinders of 16065 \* 512 = 8225280 bytes  
   
Device Boot Start End Blocks Id System  
   
/dev/sdb1 1 653 5245191 83 Linux  
   
In the above output 'sdb' is this the new LUN which was partitioned earlier?  
   
Please provide simple and detailed steps by step instruction for  
   
1) Detecting SAN device using Qlogic aswell as Emulex HBA's  
   
2) Creating Psuedo device with Powermt / Device Mapper  
  
======================================  
 RAID  
======================================  
  
  
======================================  
 Distributed Parity:  
======================================  
   
There's some confusion and different marketing terms out there. RAID 6 is a common   
 one. RAID 6 is an extension of RAID 5. RAID 5 is some number of disk drives, we'll   
 call it N + 1. N could be five drives plus a parity. Six total drives, five actual   
 data drives. The drives are striped with parity interleaved to optimize   
 performance.  
   
RAID 6 adds a second parity drive. So, think of it as N + 2. Seven total drives. The   
 need for RAID 6 is larger drives. It takes longer to rebuild larger drives when a   
 drive fails, so you have a longer exposure window if something happens. Having that  
 second parity drive protects you from an additional drive failure during the   
 rebuild. That's the basic idea behind dual parity.  
   
With distributed parity, the parity may be distributed onto another array. One way to  
 do that is with two arrays clustered together where the data is actually replicated  
 across different storage systems. It may not be parity protection, per se, but it   
 is a form of protection. There are some other techniques that involve spreading   
 parity bits across different storage systems and across wider areas. We're starting  
 to see some companies, such as Cleversafe, in the wide area distributed game, where  
 the data and parity is spread across different locations.  
   
RAID level 6 was not an original RAID level. It adds an additional parity block to a   
 RAID 5 array. It needs at least four disks (two disks for the capacity, two disks   
 for redundancy). RAID 5 can be seen as a special case of a Reed-Solomon code. RAID   
 5 is a special case, though, it only needs addition in the Galois field GF(2). This  
 is easy to do with XORs. RAID 6 extends these calculations. It is no longer a   
 special case, and all of the calculations need to be done. With RAID 6, an extra   
 checksum (called polynomial) is used, usually of GF (28). With this approach it is   
 possible to protect against any number of failed disks. RAID 6 is for the case of   
 using two checksums to protect against the loss of two disks.  
   
Like with RAID 5, parity and data are on different disks, for each block. The two   
 parity blocks are also located on different disks.  
   
RAID 6 is slower than RAID 5, but it allows the RAID to continue with any two disks   
 failed.  
  
======================================  
 PACKAGING ////////////////////////  
======================================  
  
======================================  
 RPM COMMANDS  
======================================  
  
======================================  
 Install:  
======================================  
   
# rpm -ivh foo-2.0-4.i386.rpm  
 # rpm -i ftp://ftp.redhat.com/pub/redhat/RPMS/foo-1.0-1.i386.rpm  
 # rpm -i   
 http://oss.oracle.com/projects/firewire/dist/files/kernel-2.4.20-18.10.1.i686.rpm  
 Used to install a RPM package. Note that RPM packages have file naming conventions   
 like foo-2.0-4.i386.rpm, which include the package name (foo), version (2.0),   
 release (4), and architecture (i386). Also notice that RPM understands FTP and HTTP  
 protocols for installing and querying remote RPM files.  
  
======================================  
 Uninstall:  
======================================  
   
# rpm -e foo  
 To uninstall a RPM package. Note that we used the package name foo, not the name of   
 the original package file foo-2.0-4.i386.rpm above.  
  
======================================  
 Upgrade:  
======================================  
   
# rpm -Uvh foo-1.0-2.i386.rpm  
 # rpm -Uvh ftp://ftp.redhat.com/pub/redhat/RPMS/foo-1.0-1.i386.rpm  
 # rpm -Uvh   
 http://oss.oracle.com/projects/firewire/dist/files/kernel-2.4.20-18.10.1.i686.rpm  
 To upgrade a RPM package. Using this command, RPM automatically uninstall the old   
 version of the foo package and install the new package. It is safe to always use   
 rpm -Uvh to install and upgrade packages, since it works fine even when there are   
 no previous versions of the package installed! Also notice that RPM understands FTP  
 and HTTP protocols for upgrading from remote RPM files.  
  
======================================  
 Query All:  
======================================  
   
# rpm -qa  
 To query all installed packages. This  
 command will print the names of all  
 installed packages installed on your Linux system.  
======================================  
 Query Single:  
======================================  
   
# rpm -q foo  
 To query a RPM package. This command will print the package name, version, and   
 release number of the package foo only if it is installed. Use this command to   
 verify that a package is or is not installed on your Linux system.  
======================================  
 Query Pkg. Info.:  
======================================  
   
# rpm -qi foo  
 To display package information. This command display package information including   
 the package name, version, and description of the installed program. Use this   
 command to get detailed information about the installed package.  
======================================  
 List Files in installed Pkg.:  
======================================  
   
# rpm -ql foo  
 To list files in installed package. This command will list all of files in an   
 installed RPM package. It works only when the package is already installed on your   
 Linux system.  
  
======================================  
 Which Pkg. owns a file?:  
======================================  
   
# rpm -qf /usr/bin/mysql  
 mysql-3.23.52-3  
 Which package owns a file? This command checks to determine which installed package a  
 particular file belongs to.  
  
======================================  
 List files in RPM file:  
======================================  
   
# rpm -qpl kernel-2.4.20-18.10.1.i686.rpm  
 # rpm -qpl ftp://ftp.redhat.com/pub/redhat/RPMS/foo-1.0-1.i386.rpm  
 # rpm -qpl   
 http://oss.oracle.com/projects/firewire/dist/files/kernel-2.4.20-18.10.1.i686.rpm  
 List files in RPM file. This command allows you to query a (possibly) uninstalled RPM  
 file with the use of the the "-p" option. You can use the "-p" option to operate on  
 an RPM file without actually installing anything. This command lists all files in   
 an RPM file you have in the current directory. Also note that RPM can query remote   
 files through the FTP and HTTP protocols.  
  
======================================  
 Verify Installed Package:  
======================================  
   
# rpm --verify mysql  
 To verify an installed package. This command will list all files that do NOT pass the  
 verify tests (done on size, MD5 signature, etc). Where a file does NOT pass, the   
 output is listed using the following codes that signify what failed:  
 S File size  
 M Mode (includes permissions and file type)  
 5 MD5 sum  
 L Symlink  
 D Device  
 U User  
 G Group  
 T Mtime  
 Take for example the following:  
 # rpm --verify mysql  
 S.5....T c /etc/my.cnf  
 This example indicates that file /etc/my.cnf failed on:  
 File size  
 MD5 Sum  
 Modified Time  
 However, the "c" tells us this is a configuration file so that explains the changes.   
 It should still be looked at to determine what the changes were.  
  
======================================  
 Check RPM Signature package:  
======================================  
   
# rpm --checksig foo  
 To check a RPM signature package. This command checks the PGP signature of specified   
 package to ensure its integrity and origin. Always use this command first before   
 installing a new RPM package on your system. Also, GnuPG or Pgp software must be   
 already installed on your system before you can use this command.  
  
======================================  
 Build RPM Package:  
======================================  
   
RPM package building guide  
   
   
   
Sam Isaacson ( sbi@nbcs.rutgers.edu)  
   
   
   
--------------------------------------------------------------------------------  
   
   
   
   
   
Introduction to rpm package building  
   
Rpm packages are usually built with a "spec file," which is a collection of text  
 and shell scripts that build, install and describe a software program. Here is  
 a typical spec file:  
   
   
   
Summary: Rc shell from Plan 9  
   
Name: rc  
   
Version: 1.6  
   
Release: 1  
   
Group: System Environment/Shells  
   
Copyright: BSD-type  
   
Source: rc-%{version}.tar.gz  
   
BuildRoot: %{\_tmppath}/%{name}-root  
   
Requires: readline  
   
BuildRequires: readline-devel  
   
   
   
%description  
   
rc is a command interpreter and programming language similar to sh(1).  
 It is based on the AT&T Plan 9 shell of the same name. The shell  
 offers a C-like syntax (much more so than the C shell), and a powerful  
 mechanism for manipulating variables. It is reasonably small and  
 reasonably fast, especially when compared to contemporary shells. Its  
 use is intended to be interactive, but the language lends itself well  
 to scripts.  
   
   
[from the man page]  
   
   
%prep  
   
%setup -q  
   
   
%build  
   
LD="/usr/ccs/bin/ld -L/usr/local/lib -R/usr/local/lib" \  
   
LDFLAGS="-L/usr/local/lib -R/usr/local/lib" ./configure --with-history \  
   
--with-readline  
   
make  
   
   
%install  
   
rm -rf $RPM\_BUILD\_ROOT  
   
mkdir -p $RPM\_BUILD\_ROOT/usr/local  
   
make install prefix=$RPM\_BUILD\_ROOT/usr/local sysconfdir=$RPM\_BUILD\_ROOT/etc  
   
   
%clean  
   
rm -rf $RPM\_BUILD\_ROOT  
   
   
%files  
   
%defattr(-,bin,bin)  
   
%doc COPYING AUTHORS EXAMPLES README RELDATE ChangeLog  
   
/usr/local/bin/rc  
   
/usr/local/bin/-  
   
/usr/local/bin/--  
   
/usr/local/bin/-p  
   
/usr/local/bin/--p  
   
/usr/local/man/man1/rc.1  
   
/usr/local/man/man1/history.1  
   
The spec file is split into several sections, which will be examined  
 individually.  
   
   
   
In order to write spec files, it is important to understand rpm's dependency  
 checking, macros and directory structure. When you build a package, rpm  
 creates a list of the shared libraries that it includes and a list of the  
 shared libraries to which it is linked. RPM records the shared libraries that  
 the package provides, along with the package name itself and anything manually  
 specified in the spec file, along with version information. Similarly, RPM  
 records the required shared libraries and manually specified requires. In rc,  
 readline, libc.so.1, libcurses.so.1, libdl.so.1 and libreadline.so.4 are  
 required, and rc (version 1.6) is provided. Readline and libreadline.so.4 are  
 both provided by the readline package; the rest are provided by the operating  
 system.  
   
   
   
"Build requires" -- packages required at build time -- can be specified in a  
 spec file. When it is built, rc needs readline-devel, as it has the header  
 files for the readline library.  
   
   
   
Rpm has a simple macro system. Macros can be defined like so:  
   
   
   
%define foo bar  
   
%{foo}  
   
is preprocessed to become "bar". Rpm has logical constructs: %if/%else/%endif,  
 %ifos, and %ifarch (cf. openssl.spec).  
   
   
 Finally, rpm has a system of directories for package building:  
   
   
 prefix/src/redhat/RPMS/sparc  
   
prefix/src/redhat/RPMS/sparc64  
   
prefix/src/redhat/RPMS/sparcv9  
   
.  
   
.  
   
.  
   
prefix/src/redhat/SRPMS  
   
prefix/src/redhat/SOURCES ($RPM\_SOURCE\_DIR)  
   
prefix/src/redhat/BUILD ($RPM\_BUILD\_DIR)  
   
prefix/src/redhat/SPECS  
   
RPM expects to find your source in SOURCES; it will unpack and compile the  
 source code in BUILD. RPM expects to find the files that the package will  
 install in $RPM\_BUILD\_ROOT, which rc has set to %{\_tmppath}/rc-root  
 ("%{\_tmppath}" is a macro set by rpm which expands to the name of a directory  
 for temporary files).  
   
   
   
   
   
The Preamble  
   
The preamble from rc.spec is:  
   
Summary: Rc shell from Plan 9  
   
Name: rc  
   
Version: 1.6  
   
Release: 1  
   
Group: System Environment/Shells  
   
Copyright: BSD-type  
   
Source: rc-%{version}.tar.gz  
   
BuildRoot: %{\_tmppath}/%{name}-root  
   
Requires: readline  
   
It describes the package name, version, etc. Name, Version, Release, Group,  
 Copyright (or License), and Summary are required. Other fields, such as URL  
 and Packager, are optional. Name, Version and Release define macros called  
 %{name}, %{version} and %{release} respectively.  
   
   
   
Generally, source filenames match the expansion of "%{name}-%{version}.tar.gz".  
 The %{version} macro makes maintaining the package much easier; its use is  
 highly recommended. If the source field has an URL, rpm automatically  
 downloads the source and places it in $RPM\_SOURCE\_DIR. You can specify  
 multiple sources with Source0, Source1, etc.  
   
   
   
   
   
The %description  
   
This section is parsed separately from the preamble, but can be thought of as  
 another field. Generally, one can steal the introduction from a README or man  
 page to get a good description.  
   
   
   
The %prep section  
   
   
   
%prep  
   
%setup -q  
   
The %prep section is where the source is prepared, usually in $RPM\_BUILD\_DIR.  
 Rpm provides the %setup and %patch primitives which automatically untar and  
 patch your source. %setup expects it to untar into a directory called  
 %{name}-%{version}; otherwise you have to pass it the -n switch, which renames  
 the directory. The important %setup switches are:  
   
   
   
   
   
-n <name> (name of build directory)  
   
-c (creates top-level build directory)  
   
-D (don't delete top-level build directory)  
   
-T (don't unpack Source0)  
   
-a <n> (unpack Source number n, after cd'ing to build directory)  
   
-a <n> (unpack Source number n, before cd'ing to build directory)  
   
-q (unpack silently)  
   
   
   
To unpack several sources into the same directory, you need to have something  
 like the following in %prep:  
   
   
   
%setup -q  
   
%setup -D -T -a 1  
   
%setup -D -T -a 2  
   
That unpacks source 0, then cds into %{name}-%{version} and unpacks source 1 and  
 2. As for patches, you have the following switches:  
   
   
   
-P < n > (use Patch number n)  
   
-p, -b, -E (see patch(1))  
   
   
   
While %prep appears to be all macros, don't be fooled -- %prep, %clean, %build,  
 %install, %pre, %post, etc. are all shell scripts.  
   
You might need to install GNU tar and put it on your PATH before Sun tar when  
 building packages with extremely long filenames (the GNOME software in  
 particular requires gnutar).  
   
   
   
   
The %build section  
   
%build  
   
LD="/usr/ccs/bin/ld -L/usr/local/lib -R/usr/local/lib" \  
   
LDFLAGS="-L/usr/local/lib -R/usr/local/lib" ./configure --with-history \  
   
--with-readline  
   
make  
   
The %build section is where the actual compiling takes place. Rpm has a  
 %configure macro, which is broken by design (it takes the directories from  
 prefix/lib/rpm/macros, so it might misplace your files; and it only works with  
 GNU configure). With GNU configure, you probably want to configure and build  
 the sources like so:  
   
   
   
automake # if you patched Makefile.am  
   
autoconf # if you patched configure.in  
   
LD="/usr/ccs/bin/ld -L/usr/local/lib -R/usr/local/lib" \  
   
LDFLAGS="-L/usr/local/lib -R/usr/local/lib" CPPFLAGS="-I/usr/local/include" \  
   
./configure --prefix=/usr/local --sysconfdir=/etc  
   
make  
   
Unfortunately, GNU configure may not use $LD and $LDFLAGS together -- sometimes  
 it does, and sometimes it doesn't. It is more reliable to pass everything into  
 configure (especially because it increases the chance that your specfile will  
 work on someone else's machine). If you're compiling C++ for X, add CXX="g++  
 -fpermissive" (Sun's include files aren't ANSI C++).  
   
   
   
As for imake (with Sun's cc, not gcc), try:  
   
   
   
xmkmf -a  
   
make CCOPTIONS="-I/usr/local/include" LINTOPTS="" \  
   
EXTRA\_LDOPTIONS="-L/usr/local/lib -R/usr/local/lib"  
   
Using imake and gcc is left as an exercise to the reader.  
   
   
   
Don't specify the prefix as $RPM\_BUILD\_ROOT/usr/local; many programs hardcode  
 file locations at the configure or make stage.  
   
   
   
   
The %install section  
   
%install  
   
rm -rf $RPM\_BUILD\_ROOT  
   
mkdir -p $RPM\_BUILD\_ROOT/usr/local  
   
make install prefix=$RPM\_BUILD\_ROOT/usr/local sysconfdir=$RPM\_BUILD\_ROOT/etc  
   
The %install section is where the files get "installed" into your build root.  
 You can build rpms without a build root, but this practice is highly  
 deprecated and insecure (more on this later). Always begin the %install  
 section with something along the lines of  
   
   
   
rm -rf $RPM\_BUILD\_ROOT  
   
mkdir -p $RPM\_BUILD\_ROOT/usr/local  
   
Sometimes, you can get away with just adding  
   
make install prefix=$RPM\_BUILD\_ROOT/usr/local  
   
Usually, it's a little hairier. If your program puts files in /etc, you have to  
 tell make install (if you use make install). If a program hardcodes file  
 locations at the make install stage, the best solution is to massage the  
 output of make -n install. Truly devious programs such as qmail, which compile  
 their own installer, make require patches to install correctly.  
   
   
   
   
Other scripts  
   
%clean  
   
rm -rf $RPM\_BUILD\_ROOT  
   
Generally, the only other script you need is %clean, which gets executed after  
 the build (just clean out $RPM\_BUILD\_ROOT). You also get %pre (preinstall),  
 %post (postinstall), %preun (preuninstall), %postun (postuninstall),  
 %verifyscript (executed with rpm -V), and triggers (read the documentation  
 included with rpm).  
   
   
   
   
The %files section  
   
%files  
   
%defattr(-,bin,bin)  
   
%doc COPYING AUTHORS EXAMPLES README RELDATE ChangeLog  
   
/usr/local/bin/rc  
   
/usr/local/bin/-  
   
/usr/local/bin/--  
   
/usr/local/bin/-p  
   
/usr/local/bin/--p  
   
/usr/local/man/man1/rc.1  
   
/usr/local/man/man1/history.1  
   
The %files section is where you list all the files in the package. You have a  
 few commands at your disposal: %doc (marks documentation), %attr (marks  
 attibutes of a file - mode [- means don't change mode], user, group), %defattr  
 (default attributes), %verify (see Maximum RPM), %config (marks configuration  
 files), %dir and %docdir.  
   
   
   
If a filename in the %files list corresponds to a directory, the package owns  
 the directory as well as all the files in it; so don't put /usr/bin in your  
 %files list. Be careful with globbing and directories; if you list a file  
 twice, rpm will not build your package. Also, some symlinks (absolute ones)  
 cause rpm to complain bitterly; avoid unintentionally grabbing them.  
   
   
   
   
   
Methods for generating file lists  
   
Unfortunately, generating file lists isn't always easy. Assuming that you didn't  
 have to parse the output of make -n install yourself to write the %install  
 section, try doing something sneaky like:  
   
   
   
$ ./configure --prefix=/usr/local --sysconfdir=/etc  
   
$ make  
   
$ mkdir -p sandbox/usr/local/  
   
$ make install prefix=`pwd`/sandbox/usr/local/ sysconfdir=`pwd`/etc  
   
$ for i in `find sandbox -type f`; do # check to ensure that no files  
   
> strings $i | grep sandbox && echo $i # "know" that they were installed  
   
> done # in the build root  
   
Check out the Makefile. Some packages use prefix; others use PREFIX, DESTDIR, or  
 something different. Sometimes, you don't need to add the "usr/local" part.  
 This is, incidentally, a good reason not to build packages as root&?emdash;if  
 you accidentally install the software on your system (instead of in an empty  
 directory), you cannot test your package as easily.  
   
   
   
Using the rudimentary genspec.pl script (or find(1)), you can use this directory  
 to generate a file list. After you get a list, you may wish to replace long  
 lists of files with globs. For instance:  
   
   
   
/usr/local/lib/locale/cs/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/de/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/fi/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/fr/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/ja/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/pl/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/pt\_BR/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/ru/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/sk/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/sk/LC\_MESSAGES/popt.mo  
   
/usr/local/lib/locale/sl/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/sr/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/sv/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/tr/LC\_MESSAGES/rpm.mo  
   
/usr/local/lib/locale/ro/LC\_MESSAGES/popt.mo  
   
becomes  
   
   
   
/usr/local/lib/locale/\*/LC\_MESSAGES/\*.mo  
   
This makes packages more maintainable. If Spanish translations were added, the  
 glob would catch them; otherwise, you would have to add  
 /usr/local/lib/local/es/LC\_MESSAGES/rpm.mo to the file list. You have to be  
 careful, however, that the globs catch only the files or directories you want.  
   
   
   
Sometimes, it may be appropriate to generate a file list on the fly. The perl  
 package does this:  
   
   
   
%build  
   
sh Configure -de -Dprefix=/usr/local -Dcpp='/opt/SUNWspro/bin/cc -E' \  
   
-Dcc='/opt/SUNWspro/bin/cc' \  
   
-Dinstallprefix="$RPM\_BUILD\_ROOT/usr/local" \  
   
-Dldflags='-L/usr/local/lib -R/usr/local/lib' -Dusethreads  
   
make  
   
make test  
   
   
   
%install  
   
rm -rf $RPM\_BUILD\_ROOT  
   
mkdir -p $RPM\_BUILD\_ROOT/usr/local  
   
make install  
   
   
   
# clean up files which know about the build root  
   
for fn in .packlist Config.pm; do  
   
afn="$RPM\_BUILD\_ROOT/usr/local/lib/perl5/%{version}/%{perl\_arch}/$fn"  
   
chmod 0644 $afn  
   
mv $afn $afn.TEMP  
   
sed "s#$RPM\_BUILD\_ROOT##g" < $afn.TEMP > $afn  
   
rm -f $afn.TEMP  
   
done  
   
chmod 0444 \  
   
$RPM\_BUILD\_ROOT/usr/local/lib/perl5/%{version}/%{perl\_arch}/Config.pm  
   
   
   
   
   
find $RPM\_BUILD\_ROOT -type f \( -name \\*.h -o -name \\*.a \) -print \  
   
| sed "s#^$RPM\_BUILD\_ROOT/\*#/#" > DEVEL-LIST  
   
find $RPM\_BUILD\_ROOT -type f ! \( -name \\*.h -o -name \\*.a \) -print \  
   
| sed "s#^$RPM\_BUILD\_ROOT/\*#/#" > REGULAR-LIST  
   
   
   
%files -f REGULAR-LIST  
   
%doc Copying Artistic README  
   
   
   
%files devel -f DEVEL-LIST  
   
   
   
Subpackages  
   
If you want to make more than one package out of a single source tree, you have  
 to use subpackages. Here is an example of spec file with subpackages:  
   
   
   
Name: readline  
   
Version: 4.1  
   
Copyright: GPL  
   
Group: System Environment/Libraries  
   
Summary: GNU readline  
   
Release: 1  
   
Source: readline-4.1.tar.gz  
   
Provides: libhistory.so  
   
Provides: libreadline.so  
   
BuildRoot: %{\_tmppath}/%{name}-root  
   
   
   
%description  
   
GNU readline is a library that enables history, completion, and  
   
emacs/vi-like motion functionality in a program linked with it.  
   
   
   
%package devel  
   
Summary: Readline header files, static libraries  
   
Group: Development/Libraries  
   
Requires: readline = 4.1  
   
   
   
%description devel  
   
This package contains the header files and static libraries for  
   
readline. Install this package if you want to write or compile a  
   
program that needs readline.  
   
   
   
%prep  
   
%setup -q  
   
   
   
%build  
   
autoconf  
   
LDFLAGS="-L/usr/local/lib -R/usr/local/lib" ./configure \  
   
--prefix=/usr/local --enable-shared  
   
make  
   
make shared  
   
   
   
%install  
   
rm -rf $RPM\_BUILD\_ROOT  
   
mkdir -p $RPM\_BUILD\_ROOT/usr/local  
   
make install prefix=$RPM\_BUILD\_ROOT/usr/local  
   
make install-shared prefix=$RPM\_BUILD\_ROOT/usr/local  
   
   
   
%clean  
   
rm -rf $RPM\_BUILD\_ROOT  
   
   
   
%post  
   
ln -s /usr/local//lib/libhistory.so.4 /usr/local/lib/libhistory.so  
   
ln -s /usr/local//lib/libreadline.so.4 /usr/local/lib/libreadline.so  
   
if [ -x /usr/local/bin/install-info ]; then  
   
/usr/local/bin/install-info --info-dir=/usr/local/info \  
   
/usr/local/info/rluserman.info  
   
/usr/local/bin/install-info --info-dir=/usr/local/info \  
   
/usr/local/info/history.info  
   
fi  
   
   
   
%preun  
   
rm /usr/local/lib/libhistory.so  
   
rm /usr/local/lib/libreadline.so  
   
if [ -x /usr/local/bin/install-info ]; then  
   
/usr/local/bin/install-info --delete --info-dir=/usr/local/info \  
   
/usr/local/info/rluserman.info  
   
/usr/local/bin/install-info --delete --info-dir=/usr/local/info \  
   
/usr/local/info/history.info  
   
fi  
   
   
   
%files  
   
%defattr(-,bin,bin)  
   
%doc COPYING  
   
/usr/local/lib/libhistory.so.4  
   
/usr/local/lib/libreadline.so.4  
   
/usr/local/info/readline.info  
   
/usr/local/info/rluserman.info  
   
/usr/local/info/history.info  
   
/usr/local/man/man3/readline.3  
   
   
   
%files devel  
   
%defattr(-,bin,bin)  
   
/usr/local/include/readline  
   
/usr/local/lib/libreadline.a  
   
/usr/local/lib/libhistory.a  
   
This creates two packages: readline and readline-devel. (If you just want devel,  
 replace %package devel with %package -n devel and %files with %files -n  
 devel).  
   
   
   
   
   
Style and Security  
   
Don't build packages as root; edit prefix/lib/rpm/macros so you can build in  
 your home directory. If you build as root, you run the risk of accidentally  
 installing files on your system. Instead of using chown in %install, use %attr  
 in %files.  
   
Be careful when building on a multiuser system; the buildroot, if it is in a  
 globally-writable directory, is a big security hole.  
   
Don't use the %config directive. It might break packages that a user is  
 upgrading; instead, at the end of %install, write  
   
for i in `find $RPM\_BUILD\_ROOT/etc -type f`; do  
   
mv $i $i.rpm  
   
done  
   
and warn the user in %post.  
   
Don't make the user set his or her LD\_LIBRARY\_PATH. Instead, use -R.  
   
If you need to patch configure, patch configure.in instead.  
   
Don't interactively involve the user at build or compile time.  
   
Try to split your packages into static library/header "development" packages and  
 shared library packages.  
   
If you are building GNU replacements for tools packaged with Solaris (e.g.  
 fileutils, grep, tar), put them in /usr/local/gnu instead of /usr/local. Avoid  
 putting any binaries in /usr/local/bin that conflict with any in /usr/ccs/bin,  
 /usr/bin, etc.  
   
Use %{\_tmppath} instead of /free/tmp or /var/tmp?it is more portable.  
   
   
   
More information  
   
Go to rpm.org for more information. Unfortunately, rpm is extremely poorly  
 documented. Maximum RPM is out of date; the most authoritative source on rpm  
 is rpm's source, which is kind of messy. Any one of the redhat mirrors has  
 source rpms; run them through rpm2cpio and take a look at the specfiles (which  
 are unfortunately Redhat-specific).  
   
   
   
   
   
--------------------------------------------------------------------------------  
   
$Id: guide.html,v 1.1.1.1 2001/12/14 20:38:47 sbi Exp $  
  
======================================  
 PROCESSES ////////////////////////  
======================================  
  
======================================  
 Count processes:  
======================================  
   
[ root@enterprise ]$ ps -ef | grep sendmail | sed -n '$='  
 3  
   
[ root@enterprise ]$ ps -ef |grep sendmail |wc  
 3 29 211  
   
[ root@enterprise ]$ ps -ef |grep sendmail |wc -l  
 3  
  
=======================================  
 Advanced Process Commands Solaris / Linux  
=======================================  
   
/usr/proc/bin/ptree <pid>  
   
/\* Print the parent/child process 'tree' of a process \*/  
=======================================  
   
/usr/proc/bin/pwdx <pid>  
   
/\* Print the working directory of a process \*/  
=======================================  
   
/usr/ucb/ps -aux | more  
   
/\* Displays CPU % usage for each process in ascending order \*/  
=======================================  
   
/usr/ucb/ps -auxww | grep <process name>  
   
/\* Gives the full listing of the process (long listing) \*/  
=======================================  
   
coreadm -i core.%f.%p  
   
/\* Append program name and process id to core file names \*/  
=======================================  
   
fuser -uc /var  
   
/\* Processes that are running from /var \*/  
=======================================  
   
ipcs  
   
/\* Report inter-process communication facilities status \*/  
=======================================  
   
kill -HUP `ps -ef | grep [p]roccess | awk '{print $2}'`  
   
/\* HUP any related process in one step \*/  
=======================================  
   
lsof -i TCP:25  
   
/\* Mapping port with process \*/  
=======================================  
   
pfiles <pid>  
   
/\* Shows processes' current open files \*/  
=======================================  
   
pkill -n <name>  
   
/\* Kill a process by name \*/  
=======================================  
   
prstat -a  
   
/\* An alternative for top command \*/  
=======================================  
   
ps -edf -o pcpu,pid,user,args  
   
/\* Nicely formatted 'ps' \*/  
=======================================  
   
ps -ef | grep -i <string> | awk '{ print $2 }'  
   
/\* Creates list of running PID by <string> \*/  
=======================================  
   
ps -ef | grep -i <string> | awk '{ print $2 }'  
   
/\* Creates list of running PID by \*/  
=======================================  
   
ps -ef | grep <process name> | grep -v grep | cut -c 10-15 | xargs kill -9  
   
/\* Find and kill all instances of a given process \*/  
=======================================  
   
ps -ef | more  
   
/\* Show all processes running \*/  
=======================================  
   
ps -ef|grep -v "0:00"|more  
   
/\* Gives you a list of any process with CPU time more than 0:00 \*/  
=======================================  
   
ps -eo pid,args  
   
/\* List processes in simplified format \*/  
=======================================  
   
ps -fu oracle|grep pmon  
   
/\* See which instances of Oracle are running \*/  
=======================================  
   
top -b 1  
   
/\* Returns the process utilizing the most cpu and quits \*/  
=======================================  
  
======================================  
 SYSTEM INTERNALS /////////////////  
======================================  
  
======================================  
 SYSTEM INTERNALS LINUX:  
======================================  
   
Every process under Linux is dynamically allocated a struct task\_struct structure.   
 The maximum number of processes which can be created on Linux is limited only by   
 the amount of physical memory present, and is equal to (see   
 kernel/fork.c:fork\_init()):  
   
/\*  
 \* The default maximum number of threads is set to a safe  
 \* value: the thread structures can take up at most half  
 \* of memory.  
 \*/  
 max\_threads = mempages / (THREAD\_SIZE/PAGE\_SIZE) / 2;  
   
which, on IA32 architecture, basically means num\_physpages/4. As an example, on a   
 512M machine, you can create 32k threads. This is a considerable improvement over   
 the 4k-epsilon limit for older (2.2 and earlier) kernels. Moreover, this can be   
 changed at runtime using the KERN\_MAX\_THREADS sysctl(2), or simply using procfs   
 interface to kernel tunables:  
   
# cat /proc/sys/kernel/threads-max  
 32764  
 # echo 100000 > /proc/sys/kernel/threads-max  
 # cat /proc/sys/kernel/threads-max  
 100000  
 # gdb -q vmlinux /proc/kcore  
 Core was generated by `BOOT\_IMAGE=240ac18 ro root=306 video=matrox:vesa:0x118'.  
 #0 0x0 in ?? ()  
 (gdb) p max\_threads  
 $1 = 100000  
   
The set of processes on the Linux system is represented as a collection of struct   
 task\_struct structures which are linked in two ways:  
   
1. as a hashtable, hashed by pid, and  
 2. as a circular, doubly-linked list using p->next\_task and p->prev\_task pointers.  
   
The hashtable is called pidhash[] and is defined in include/linux/sched.h:  
   
/\* PID hashing. (shouldnt this be dynamic?) \*/  
 #define PIDHASH\_SZ (4096 >> 2)  
 extern struct task\_struct \*pidhash[PIDHASH\_SZ];  
   
#define pid\_hashfn(x) ((((x) >> 8) ^ (x)) & (PIDHASH\_SZ - 1))  
   
The tasks are hashed by their pid value and the above hashing function is supposed to  
 distribute the elements uniformly in their domain (0 to PID\_MAX-1). The hashtable   
 is used to quickly find a task by given pid, using find\_task\_pid() inline from   
 include/linux/sched.h:  
   
static inline struct task\_struct \*find\_task\_by\_pid(int pid)  
 {  
 struct task\_struct \*p, \*\*htable = &pidhash[pid\_hashfn(pid)];  
   
for(p = \*htable; p && p->pid != pid; p = p->pidhash\_next)  
 ;  
   
return p;  
 }  
   
The tasks on each hashlist (i.e. hashed to the same value) are linked by p->  
 pidhash\_next/pidhash\_pprev which are used by hash\_pid() and unhash\_pid() to insert   
 and remove a given process into the hashtable. These are done under protection of   
 the read-write spinlock called tasklist\_lock taken for WRITE.  
   
The circular doubly-linked list that uses p->next\_task/prev\_task is maintained so   
 that one could go through all tasks on the system easily. This is achieved by the   
 for\_each\_task() macro from include/linux/sched.h:  
   
#define for\_each\_task(p) \  
 for (p = &init\_task ; (p = p->next\_task) != &init\_task ; )  
   
Users of for\_each\_task() should take tasklist\_lock for READ. Note that   
 for\_each\_task() is using init\_task to mark the beginning (and end) of the list -   
 this is safe because the idle task (pid 0) never exits.  
   
The modifiers of the process hashtable or/and the process table links, notably   
 fork(), exit() and ptrace(), must take tasklist\_lock for WRITE. What is more   
 interesting is that the writers must also disable interrupts on the local CPU. The   
 reason for this is not trivial: the send\_sigio() function walks the task list and   
 thus takes tasklist\_lock for READ, and it is called from kill\_fasync() in interrupt  
 context. This is why writers must disable interrupts while readers don't need to.  
   
Now that we understand how the task\_struct structures are linked together, let us   
 examine the members of task\_struct. They loosely correspond to the members of UNIX   
 'struct proc' and 'struct user' combined together.  
   
The other versions of UNIX separated the task state information into one part which   
 should be kept memory-resident at all times (called 'proc structure' which includes  
 process state, scheduling information etc.) and another part which is only needed   
 when the process is running (called 'u area' which includes file descriptor table,   
 disk quota information etc.). The only reason for such ugly design was that memory   
 was a very scarce resource. Modern operating systems (well, only Linux at the   
 moment but others, e.g. FreeBSD seem to improve in this direction towards Linux) do  
 not need such separation and therefore maintain process state in a kernel   
 memory-resident data structure at all times.  
   
The task\_struct structure is declared in include/linux/sched.h and is currently 1680   
 bytes in size.  
   
The state field is declared as:  
   
volatile long state; /\* -1 unrunnable, 0 runnable, >0 stopped \*/  
   
#define TASK\_RUNNING 0  
 #define TASK\_INTERRUPTIBLE 1  
 #define TASK\_UNINTERRUPTIBLE 2  
 #define TASK\_ZOMBIE 4  
 #define TASK\_STOPPED 8  
 #define TASK\_EXCLUSIVE 32  
   
Why is TASK\_EXCLUSIVE defined as 32 and not 16? Because 16 was used up by   
 TASK\_SWAPPING and I forgot to shift TASK\_EXCLUSIVE up when I removed all references  
 to TASK\_SWAPPING (sometime in 2.3.x).  
   
The volatile in p->state declaration means it can be modified asynchronously (from   
 interrupt handler):  
   
1. TASK\_RUNNING: means the task is "supposed to be" on the run queue. The reason it   
 may not yet be on the runqueue is that marking a task as TASK\_RUNNING and placing   
 it on the runqueue is not atomic. You need to hold the runqueue\_lock read-write   
 spinlock for read in order to look at the runqueue. If you do so, you will then see  
 that every task on the runqueue is in TASK\_RUNNING state. However, the converse is   
 not true for the reason explained above. Similarly, drivers can mark themselves (or  
 rather the process context they run in) as TASK\_INTERRUPTIBLE (or   
 TASK\_UNINTERRUPTIBLE) and then call schedule(), which will then remove it from the   
 runqueue (unless there is a pending signal, in which case it is left on the   
 runqueue).  
  
 2. TASK\_INTERRUPTIBLE: means the task is sleeping but can be woken up by a signal or   
 by expiry of a timer.  
  
 3. TASK\_UNINTERRUPTIBLE: same as TASK\_INTERRUPTIBLE, except it cannot be woken up.  
  
 4. TASK\_ZOMBIE: task has terminated but has not had its status collected (wait()-ed   
 for) by the parent (natural or by adoption).  
  
 5. TASK\_STOPPED: task was stopped, either due to job control signals or due to   
 ptrace(2).  
  
 6. TASK\_EXCLUSIVE: this is not a separate state but can be OR-ed to either one of   
 TASK\_INTERRUPTIBLE or TASK\_UNINTERRUPTIBLE. This means that when this task is   
 sleeping on a wait queue with many other tasks, it will be woken up alone instead   
 of causing "thundering herd" problem by waking up all the waiters.  
   
Task flags contain information about the process states which are not mutually   
 exclusive:  
   
unsigned long flags; /\* per process flags, defined below \*/  
 /\* Per process flags \*/  
 #define PF\_ALIGNWARN 0x00000001 /\* Print alignment warning msgs \*/  
 /\* Not implemented yet, only for 486 \*/  
 #define PF\_STARTING 0x00000002 /\* being created \*/  
 #define PF\_EXITING 0x00000004 /\* getting shut down \*/  
 #define PF\_FORKNOEXEC 0x00000040 /\* forked but didn't exec \*/  
 #define PF\_SUPERPRIV 0x00000100 /\* used super-user privileges \*/  
 #define PF\_DUMPCORE 0x00000200 /\* dumped core \*/  
 #define PF\_SIGNALED 0x00000400 /\* killed by a signal \*/  
 #define PF\_MEMALLOC 0x00000800 /\* Allocating memory \*/  
 #define PF\_VFORK 0x00001000 /\* Wake up parent in mm\_release \*/  
 #define PF\_USEDFPU 0x00100000 /\* task used FPU this quantum (SMP) \*/  
   
The fields p->has\_cpu, p->processor, p->counter, p->priority, p->policy and p->  
 rt\_priority are related to the scheduler and will be looked at later.  
   
The fields p->mm and p->active\_mm point respectively to the process' address space   
 described by mm\_struct structure and to the active address space if the process   
 doesn't have a real one (e.g. kernel threads). This helps minimise TLB flushes on   
 switching address spaces when the task is scheduled out. So, if we are   
 scheduling-in the kernel thread (which has no p->mm) then its next->active\_mm will   
 be set to the prev->active\_mm of the task that was scheduled-out, which will be the  
 same as prev->mm if prev->mm != NULL. The address space can be shared between   
 threads if CLONE\_VM flag is passed to the clone(2) system call or by means of   
 vfork(2) system call.  
   
The fields p->exec\_domain and p->personality relate to the personality of the task,   
 i.e. to the way certain system calls behave in order to emulate the "personality"   
 of foreign flavours of UNIX.  
   
The field p->fs contains filesystem information, which under Linux means three pieces  
 of information:  
   
1. root directory's dentry and mountpoint,  
 2. alternate root directory's dentry and mountpoint,  
 3. current working directory's dentry and mountpoint.  
   
This structure also includes a reference count because it can be shared between   
 cloned tasks when CLONE\_FS flag is passed to the clone(2) system call.  
   
The field p->files contains the file descriptor table. This too can be shared between  
 tasks, provided CLONE\_FILES is specified with clone(2) system call.  
   
The field p->sig contains signal handlers and can be shared between cloned tasks by   
 means of CLONE\_SIGHAND.  
   
2.2 Creation and termination of tasks and kernel threads  
   
Different books on operating systems define a "process" in different ways, starting   
 from "instance of a program in execution" and ending with "that which is produced   
 by clone(2) or fork(2) system calls". Under Linux, there are three kinds of   
 processes:  
   
\* the idle thread(s),  
 \* kernel threads,  
 \* user tasks.  
   
The idle thread is created at compile time for the first CPU; it is then "manually"   
 created for each CPU by means of arch-specific fork\_by\_hand() in   
 arch/i386/kernel/smpboot.c, which unrolls the fork(2) system call by hand (on some   
 archs). Idle tasks share one init\_task structure but have a private TSS structure,   
 in the per-CPU array init\_tss. Idle tasks all have pid = 0 and no other task can   
 share pid, i.e. use CLONE\_PID flag to clone(2).  
   
Kernel threads are created using kernel\_thread() function which invokes the clone(2)   
 system call in kernel mode. Kernel threads usually have no user address space, i.e.  
 p->mm = NULL, because they explicitly do exit\_mm(), e.g. via daemonize() function.   
 Kernel threads can always access kernel address space directly. They are allocated   
 pid numbers in the low range. Running at processor's ring 0 (on x86, that is)   
 implies that the kernel threads enjoy all I/O privileges and cannot be pre-empted   
 by the scheduler.  
   
User tasks are created by means of clone(2) or fork(2) system calls, both of which   
 internally invoke kernel/fork.c:do\_fork().  
   
Let us understand what happens when a user process makes a fork(2) system call.   
 Although fork(2) is architecture-dependent due to the different ways of passing   
 user stack and registers, the actual underlying function do\_fork() that does the   
 job is portable and is located at kernel/fork.c.  
   
The following steps are done:  
   
1. Local variable retval is set to -ENOMEM, as this is the value which errno should   
 be set to if fork(2) fails to allocate a new task structure.  
  
 2. If CLONE\_PID is set in clone\_flags then return an error (-EPERM), unless the   
 caller is the idle thread (during boot only). So, normal user threads cannot pass   
 CLONE\_PID to clone(2) and expect it to succeed. For fork(2), this is irrelevant as   
 clone\_flags is set to SIFCHLD - this is only relevant when do\_fork() is invoked   
 from sys\_clone() which passes the clone\_flags from the value requested from   
 userspace.  
  
 3. current->vfork\_sem is initialised (it is later cleared in the child). This is used  
 by sys\_vfork() (vfork(2) system call, corresponds to clone\_flags =   
 CLONE\_VFORK|CLONE\_VM|SIGCHLD) to make the parent sleep until the child does   
 mm\_release(), for example as a result of exec()ing another program or exit(2)-ing.  
  
 4. A new task structure is allocated using arch-dependent alloc\_task\_struct() macro.   
 On x86 it is just a gfp at GFP\_KERNEL priority. This is the first reason why   
 fork(2) system call may sleep. If this allocation fails, we return -ENOMEM.  
  
 5. All the values from current process' task structure are copied into the new one,   
 using structure assignment \*p = \*current. Perhaps this should be replaced by a   
 memcpy? Later on, the fields that should not be inherited by the child are set to   
 the correct values.  
  
 6. Big kernel lock is taken as the rest of the code would otherwise be   
 non-reentrant.  
  
 7. If the parent has user resources (a concept of UID, Linux is flexible enough to   
 make it a question rather than a fact), then verify if the user exceeded   
 RLIMIT\_NPROC soft limit - if so, fail with -EAGAIN, if not, increment the count of   
 processes by given uid p->user->count.  
  
 8. If the system-wide number of tasks exceeds the value of the tunable max\_threads,   
 fail with -EAGAIN.  
  
 9. If the binary being executed belongs to a modularised execution domain, increment   
 the corresponding module's reference count.  
  
 10. If the binary being executed belongs to a modularised binary format, increment   
 the corresponding module's reference count.  
  
 11. The child is marked as 'has not execed' (p->did\_exec = 0)  
  
 12. The child is marked as 'not-swappable' (p->swappable = 0)  
  
 13. The child is put into 'uninterruptible sleep' state, i.e. p->state =   
 TASK\_UNINTERRUPTIBLE (TODO: why is this done? I think it's not needed - get rid of   
 it, Linus confirms it is not needed)  
  
 14. The child's p->flags are set according to the value of clone\_flags; for plain   
 fork(2), this will be p->flags = PF\_FORKNOEXEC.  
  
 15. The child's pid p->pid is set using the fast algorithm in kernel/fork.c:get\_pid()  
 (TODO: lastpid\_lock spinlock can be made redundant since get\_pid() is always called  
 under big kernel lock from do\_fork(), also remove flags argument of get\_pid(),   
 patch sent to Alan on 20/06/2000 - followup later).  
  
 16. The rest of the code in do\_fork() initialises the rest of child's task structure.  
 At the very end, the child's task structure is hashed into the pidhash hashtable   
 and the child is woken up (TODO: wake\_up\_process(p) sets p->state = TASK\_RUNNING   
 and adds the process to the runq, therefore we probably didn't need to set p->state  
 to TASK\_RUNNING earlier on in do\_fork()). The interesting part is setting p->  
 exit\_signal to clone\_flags & CSIGNAL, which for fork(2) means just SIGCHLD and   
 setting p->pdeath\_signal to 0. The pdeath\_signal is used when a process 'forgets'   
 the original parent (by dying) and can be set/get by means of PR\_GET/SET\_PDEATHSIG   
 commands of prctl(2) system call (You might argue that the way the value of   
 pdeath\_signal is returned via userspace pointer argument in prctl(2) is a bit silly  
 - mea culpa, after Andries Brouwer updated the manpage it was too late to fix ;)  
   
Thus tasks are created. There are several ways for tasks to terminate:  
   
1. by making exit(2) system call;  
 2. by being delivered a signal with default disposition to die;  
 3. by being forced to die under certain exceptions;  
 4. by calling bdflush(2) with func == 1 (this is Linux-specific, for compatibility   
 with old distributions that still had the 'update' line in /etc/inittab - nowadays   
 the work of update is done by kernel thread kupdate).  
   
Functions implementing system calls under Linux are prefixed with sys\_, but they are   
 usually concerned only with argument checking or arch-specific ways to pass some   
 information and the actual work is done by do\_ functions. So it is with sys\_exit()   
 which calls do\_exit() to do the work. Although, other parts of the kernel sometimes  
 invoke sys\_exit() while they should really call do\_exit().  
   
The function do\_exit() is found in kernel/exit.c. The points to note about   
 do\_exit():  
   
\* Uses global kernel lock (locks but doesn't unlock).  
 \* Calls schedule() at the end, which never returns.  
 \* Sets the task state to TASK\_ZOMBIE.  
 \* Notifies any child with current->pdeath\_signal, if not 0.  
 \* Notifies the parent with a current->exit\_signal, which is usually equal to   
 SIGCHLD.  
 \* Releases resources allocated by fork, closes open files etc.  
 \* On architectures that use lazy FPU switching (ia64, mips, mips64) (TODO: remove   
 'flags' argument of sparc, sparc64), do whatever the hardware requires to pass the   
 FPU ownership (if owned by current) to "none".  
   
2.3 Linux Scheduler  
   
The job of a scheduler is to arbitrate access to the current CPU between multiple   
 processes. The scheduler is implemented in the 'main kernel file' kernel/sched.c.   
 The corresponding header file include/linux/sched.h is included (either explicitly   
 or indirectly) by virtually every kernel source file.  
   
The fields of task structure relevant to scheduler include:  
   
\* p->need\_resched: this field is set if schedule() should be invoked at the 'next   
 opportunity'.  
 \* p->counter: number of clock ticks left to run in this scheduling slice, decremented  
 by a timer. When this field becomes lower than or equal to zero, it is reset to 0   
 and p->need\_resched is set. This is also sometimes called 'dynamic priority' of a   
 process because it can change by itself.  
 \* p->priority: the process' static priority, only changed through well-known system   
 calls like nice(2), POSIX.1b sched\_setparam(2) or 4.4BSD/SVR4 setpriority(2).  
 \* p->rt\_priority: realtime priority  
 \* p->policy: the scheduling policy, specifies which scheduling class the task belongs  
 to. Tasks can change their scheduling class using the sched\_setscheduler(2) system   
 call. The valid values are SCHED\_OTHER (traditional UNIX process), SCHED\_FIFO   
 (POSIX.1b FIFO realtime process) and SCHED\_RR (POSIX round-robin realtime process).  
 One can also OR SCHED\_YIELD to any of these values to signify that the process   
 decided to yield the CPU, for example by calling sched\_yield(2) system call. A FIFO  
 realtime process will run until either a) it blocks on I/O, b) it explicitly yields  
 the CPU or c) it is preempted by another realtime process with a higher p->  
 rt\_priority value. SCHED\_RR is the same as SCHED\_FIFO, except that when its   
 timeslice expires it goes back to the end of the runqueue.  
   
The scheduler's algorithm is simple, despite the great apparent complexity of the   
 schedule() function. The function is complex because it implements three scheduling  
 algorithms in one and also because of the subtle SMP-specifics.  
   
The apparently 'useless' gotos in schedule() are there for a purpose - to generate   
 the best optimised (for i386) code. Also, note that scheduler (like most of the   
 kernel) was completely rewritten for 2.4, therefore the discussion below does not   
 apply to 2.2 or earlier kernels.  
   
Let us look at the function in detail:  
   
1. If current->active\_mm == NULL then something is wrong. Current process, even a   
 kernel thread (current->mm == NULL) must have a valid p->active\_mm at all times.  
  
 2. If there is something to do on the tq\_scheduler task queue, process it now. Task   
 queues provide a kernel mechanism to schedule execution of functions at a later   
 time. We shall look at it in details elsewhere.  
  
 3. Initialise local variables prev and this\_cpu to current task and current CPU   
 respectively.  
  
 4. Check if schedule() was invoked from interrupt handler (due to a bug) and panic if  
 so.  
  
 5. Release the global kernel lock.  
  
 6. If there is some work to do via softirq mechanism, do it now.  
  
 7. Initialise local pointer struct schedule\_data \*sched\_data to point to per-CPU   
 (cacheline-aligned to prevent cacheline ping-pong) scheduling data area, which   
 contains the TSC value of last\_schedule and the pointer to last scheduled task   
 structure (TODO: sched\_data is used on SMP only but why does init\_idle()   
 initialises it on UP as well?).  
  
 8. runqueue\_lock spinlock is taken. Note that we use spin\_lock\_irq() because in   
 schedule() we guarantee that interrupts are enabled. Therefore, when we unlock   
 runqueue\_lock, we can just re-enable them instead of saving/restoring eflags   
 (spin\_lock\_irqsave/restore variant).  
  
 9. task state machine: if the task is in TASK\_RUNNING state, it is left alone; if it   
 is in TASK\_INTERRUPTIBLE state and a signal is pending, it is moved into   
 TASK\_RUNNING state. In all other cases, it is deleted from the runqueue.  
  
 10. next (best candidate to be scheduled) is set to the idle task of this cpu.   
 However, the goodness of this candidate is set to a very low value (-1000), in hope  
 that there is someone better than that.  
  
 11. If the prev (current) task is in TASK\_RUNNING state, then the current goodness is  
 set to its goodness and it is marked as a better candidate to be scheduled than the  
 idle task.  
  
 12. Now the runqueue is examined and a goodness of each process that can be scheduled  
 on this cpu is compared with current value; the process with highest goodness wins.  
 Now the concept of "can be scheduled on this cpu" must be clarified: on UP, every   
 process on the runqueue is eligible to be scheduled; on SMP, only process not   
 already running on another cpu is eligible to be scheduled on this cpu. The   
 goodness is calculated by a function called goodness(), which treats realtime   
 processes by making their goodness very high (1000 + p->rt\_priority), this being   
 greater than 1000 guarantees that no SCHED\_OTHER process can win; so they only   
 contend with other realtime processes that may have a greater p->rt\_priority. The   
 goodness function returns 0 if the process' time slice (p->counter) is over. For   
 non-realtime processes, the initial value of goodness is set to p->counter - this   
 way, the process is less likely to get CPU if it already had it for a while, i.e.   
 interactive processes are favoured more than CPU bound number crunchers. The   
 arch-specific constant PROC\_CHANGE\_PENALTY attempts to implement "cpu affinity"   
 (i.e. give advantage to a process on the same CPU). It also gives a slight   
 advantage to processes with mm pointing to current active\_mm or to processes with   
 no (user) address space, i.e. kernel threads.  
  
 13. if the current value of goodness is 0 then the entire list of processes (not just  
 the ones on the runqueue!) is examined and their dynamic priorities are   
 recalculated using simple algorithm:  
   
recalculate:  
 {  
 struct task\_struct \*p;  
 spin\_unlock\_irq(&runqueue\_lock);  
 read\_lock(&tasklist\_lock);  
 for\_each\_task(p)  
 p->counter = (p->counter >> 1) + p->priority;  
 read\_unlock(&tasklist\_lock);  
 spin\_lock\_irq(&runqueue\_lock);  
 }  
   
Note that the we drop the runqueue\_lock before we recalculate. The reason is that we   
 go through entire set of processes; this can take a long time, during which the   
 schedule() could be called on another CPU and select a process with goodness good   
 enough for that CPU, whilst we on this CPU were forced to recalculate. Ok,   
 admittedly this is somewhat inconsistent because while we (on this CPU) are   
 selecting a process with the best goodness, schedule() running on another CPU could  
 be recalculating dynamic priorities.  
  
 14. From this point on it is certain that next points to the task to be scheduled, so  
 we initialise next->has\_cpu to 1 and next->processor to this\_cpu. The runqueue\_lock  
 can now be unlocked.  
  
 15. If we are switching back to the same task (next == prev) then we can simply   
 reacquire the global kernel lock and return, i.e. skip all the hardware-level   
 (registers, stack etc.) and VM-related (switch page directory, recalculate   
 active\_mm etc.) stuff.  
  
 16. The macro switch\_to() is architecture specific. On i386, it is concerned with a)   
 FPU handling, b) LDT handling, c) reloading segment registers, d) TSS handling and   
 e) reloading debug registers.  
   
2.4 Linux linked list implementation  
   
Before we go on to examine implementation of wait queues, we must acquaint ourselves   
 with the Linux standard doubly-linked list implementation. Wait queues (as well as   
 everything else in Linux) make heavy use of them and they are called in jargon   
 "list.h implementation" because the most relevant file is include/linux/list.h.  
   
The fundamental data structure here is struct list\_head:  
   
struct list\_head {  
 struct list\_head \*next, \*prev;  
 };  
   
#define LIST\_HEAD\_INIT(name) { &(name), &(name) }  
   
#define LIST\_HEAD(name) \  
 struct list\_head name = LIST\_HEAD\_INIT(name)  
   
#define INIT\_LIST\_HEAD(ptr) do { \  
 (ptr)->next = (ptr); (ptr)->prev = (ptr); \  
 } while (0)  
   
#define list\_entry(ptr, type, member) \  
 ((type \*)((char \*)(ptr)-(unsigned long)(&((type \*)0)->member)))  
   
#define list\_for\_each(pos, head) \  
 for (pos = (head)->next; pos != (head); pos = pos->next)  
   
The first three macros are for initialising an empty list by pointing both next and   
 prev pointers to itself. It is obvious from C syntactical restrictions which ones   
 should be used where - for example, LIST\_HEAD\_INIT() can be used for structure's   
 element initialisation in declaration, the second can be used for static variable   
 initialising declarations and the third can be used inside a function.  
   
The macro list\_entry() gives access to individual list element, for example (from   
 fs/file\_table.c:fs\_may\_remount\_ro()):  
   
struct super\_block {  
 ...  
 struct list\_head s\_files;  
 ...  
 } \*sb = &some\_super\_block;  
   
struct file {  
 ...  
 struct list\_head f\_list;  
 ...  
 } \*file;  
   
struct list\_head \*p;  
   
for (p = sb->s\_files.next; p != &sb->s\_files; p = p->next) {  
 struct file \*file = list\_entry(p, struct file, f\_list);  
 do something to 'file'  
 }  
   
A good example of the use of list\_for\_each() macro is in the scheduler where we walk   
 the runqueue looking for the process with highest goodness:  
   
static LIST\_HEAD(runqueue\_head);  
 struct list\_head \*tmp;  
 struct task\_struct \*p;  
   
list\_for\_each(tmp, &runqueue\_head) {  
 p = list\_entry(tmp, struct task\_struct, run\_list);  
 if (can\_schedule(p)) {  
 int weight = goodness(p, this\_cpu, prev->active\_mm);  
 if (weight > c)  
 c = weight, next = p;  
 }  
 }  
   
Here, p->run\_list is declared as struct list\_head run\_list inside task\_struct   
 structure and serves as anchor to the list. Removing an element from the list and   
 adding (to head or tail of the list) is done by   
 list\_del()/list\_add()/list\_add\_tail() macros. The examples below are adding and   
 removing a task from runqueue:  
   
static inline void del\_from\_runqueue(struct task\_struct \* p)  
 {  
 nr\_running--;  
 list\_del(&p->run\_list);  
 p->run\_list.next = NULL;  
 }  
   
static inline void add\_to\_runqueue(struct task\_struct \* p)  
 {  
 list\_add(&p->run\_list, &runqueue\_head);  
 nr\_running++;  
 }  
   
static inline void move\_last\_runqueue(struct task\_struct \* p)  
 {  
 list\_del(&p->run\_list);  
 list\_add\_tail(&p->run\_list, &runqueue\_head);  
 }  
   
static inline void move\_first\_runqueue(struct task\_struct \* p)  
 {  
 list\_del(&p->run\_list);  
 list\_add(&p->run\_list, &runqueue\_head);  
 }  
   
2.5 Wait Queues  
   
When a process requests the kernel to do something which is currently impossible but   
 that may become possible later, the process is put to sleep and is woken up when   
 the request is more likely to be satisfied. One of the kernel mechanisms used for   
 this is called a 'wait queue'.  
   
Linux implementation allows wake-on semantics using TASK\_EXCLUSIVE flag. With   
 waitqueues, you can either use a well-known queue and then simply   
 sleep\_on/sleep\_on\_timeout/interruptible\_sleep\_on/interruptible\_sleep\_on\_timeout, or  
 you can define your own waitqueue and use add/remove\_wait\_queue to add and remove   
 yourself from it and wake\_up/wake\_up\_interruptible to wake up when needed.  
   
An example of the first usage of waitqueues is interaction between the page allocator  
 (in mm/page\_alloc.c:\_\_alloc\_pages()) and the kswapd kernel daemon (in   
 mm/vmscan.c:kswap()), by means of wait queue kswapd\_wait, declared in mm/vmscan.c;   
 the kswapd daemon sleeps on this queue, and it is woken up whenever the page   
 allocator needs to free up some pages.  
   
An example of autonomous waitqueue usage is interaction between user process   
 requesting data via read(2) system call and kernel running in the interrupt context  
 to supply the data. An interrupt handler might look like (simplified   
 drivers/char/rtc\_interrupt()):  
   
static DECLARE\_WAIT\_QUEUE\_HEAD(rtc\_wait);  
   
void rtc\_interrupt(int irq, void \*dev\_id, struct pt\_regs \*regs)  
 {  
 spin\_lock(&rtc\_lock);  
 rtc\_irq\_data = CMOS\_READ(RTC\_INTR\_FLAGS);  
 spin\_unlock(&rtc\_lock);  
 wake\_up\_interruptible(&rtc\_wait);  
 }  
   
So, the interrupt handler obtains the data by reading from some device-specific I/O   
 port (CMOS\_READ() macro turns into a couple outb/inb) and then wakes up whoever is   
 sleeping on the rtc\_wait wait queue.  
   
Now, the read(2) system call could be implemented as:  
   
ssize\_t rtc\_read(struct file file, char \*buf, size\_t count, loff\_t \*ppos)  
 {  
 DECLARE\_WAITQUEUE(wait, current);  
 unsigned long data;  
 ssize\_t retval;  
   
add\_wait\_queue(&rtc\_wait, &wait);  
 current->state = TASK\_INTERRUPTIBLE;  
 do {  
 spin\_lock\_irq(&rtc\_lock);  
 data = rtc\_irq\_data;  
 rtc\_irq\_data = 0;  
 spin\_unlock\_irq(&rtc\_lock);  
   
if (data != 0)  
 break;  
   
if (file->f\_flags & O\_NONBLOCK) {  
 retval = -EAGAIN;  
 goto out;  
 }  
 if (signal\_pending(current)) {  
 retval = -ERESTARTSYS;  
 goto out;  
 }  
 schedule();  
 } while(1);  
 retval = put\_user(data, (unsigned long \*)buf);  
 if (!retval)  
 retval = sizeof(unsigned long);  
   
out:  
 current->state = TASK\_RUNNING;  
 remove\_wait\_queue(&rtc\_wait, &wait);  
 return retval;  
 }  
   
What happens in rtc\_read() is this:  
   
 1. We declare a wait queue element pointing to current process context.  
   
 2. We add this element to the rtc\_wait waitqueue.  
   
 3. We mark current context as TASK\_INTERRUPTIBLE which means it will not be   
 rescheduled after the next time it sleeps.  
   
 4. We check if there is no data available; if there is we break out, copy data to   
 user buffer, mark ourselves as TASK\_RUNNING, remove ourselves from the wait queue   
 and return  
 5. If there is no data yet, we check whether the user specified non-blocking I/O and   
 if so we fail with EAGAIN (which is the same as EWOULDBLOCK)  
  
 6. We also check if a signal is pending and if so inform the "higher layers" to   
 restart the system call if necessary. By "if necessary" I meant the details of   
 signal disposition as specified in sigaction(2) system call.  
  
 7. Then we "switch out", i.e. fall asleep, until woken up by the interrupt handler.   
 If we didn't mark ourselves as TASK\_INTERRUPTIBLE then the scheduler could schedule  
 us sooner than when the data is available, thus causing unneeded processing.  
   
It is also worth pointing out that, using wait queues, it is rather easy to implement  
 the poll(2) system call:  
   
static unsigned int rtc\_poll(struct file \*file, poll\_table \*wait)  
 {  
 unsigned long l;  
   
poll\_wait(file, &rtc\_wait, wait);  
   
spin\_lock\_irq(&rtc\_lock);  
 l = rtc\_irq\_data;  
 spin\_unlock\_irq(&rtc\_lock);  
   
if (l != 0)  
 return POLLIN | POLLRDNORM;  
 return 0;  
 }  
   
All the work is done by the device-independent function poll\_wait() which does the   
 necessary waitqueue manipulations; all we need to do is point it to the waitqueue   
 which is woken up by our device-specific interrupt handler.  
   
2.6 Kernel Timers  
   
Now let us turn our attention to kernel timers. Kernel timers are used to dispatch   
 execution of a particular function (called 'timer handler') at a specified time in   
 the future. The main data structure is struct timer\_list declared in   
 include/linux/timer.h:  
   
struct timer\_list {  
 struct list\_head list;  
 unsigned long expires;  
 unsigned long data;  
 void (\*function)(unsigned long);  
 volatile int running;  
 };  
   
The list field is for linking into the internal list, protected by the timerlist\_lock  
 spinlock. The expires field is the value of jiffies when the function handler   
 should be invoked with data passed as a parameter. The running field is used on SMP  
 to test if the timer handler is currently running on another CPU.  
   
The functions add\_timer() and del\_timer() add and remove a given timer to the list.   
 When a timer expires, it is removed automatically. Before a timer is used, it MUST   
 be initialised by means of init\_timer() function. And before it is added, the   
 fields function and expires must be set.  
   
2.7 Bottom Halves  
   
Sometimes it is reasonable to split the amount of work to be performed inside an   
 interrupt handler into immediate work (e.g. acknowledging the interrupt, updating   
 the stats etc.) and work which can be postponed until later, when interrupts are   
 enabled (e.g. to do some postprocessing on data, wake up processes waiting for this  
 data, etc).  
   
Bottom halves are the oldest mechanism for deferred execution of kernel tasks and   
 have been available since Linux 1.x. In Linux 2.0, a new mechanism was added,   
 called 'task queues', which will be the subject of next section.  
   
Bottom halves are serialised by the global\_bh\_lock spinlock, i.e. there can only be   
 one bottom half running on any CPU at a time. However, when attempting to execute   
 the handler, if global\_bh\_lock is not available, the bottom half is marked (i.e.   
 scheduled) for execution - so processing can continue, as opposed to a busy loop on  
 global\_bh\_lock.  
   
There can only be 32 bottom halves registered in total. The functions required to   
 manipulate bottom halves are as follows (all exported to modules):  
   
\* void init\_bh(int nr, void (\*routine)(void)): installs a bottom half handler pointed  
 to by routine argument into slot nr. The slot ought to be enumerated in   
 include/linux/interrupt.h in the form XXXX\_BH, e.g. TIMER\_BH or TQUEUE\_BH.   
 Typically, a subsystem's initialisation routine (init\_module() for modules)   
 installs the required bottom half using this function.  
   
\* void remove\_bh(int nr): does the opposite of init\_bh(), i.e. de-installs bottom   
 half installed at slot nr. There is no error checking performed there, so, for   
 example remove\_bh(32) will panic/oops the system. Typically, a subsystem's cleanup   
 routine (cleanup\_module() for modules) uses this function to free up the slot that   
 can later be reused by some other subsystem. (TODO: wouldn't it be nice to have   
 /proc/bottom\_halves list all registered bottom halves on the system? That means   
 global\_bh\_lock must be made read/write, obviously)  
   
\* void mark\_bh(int nr): marks bottom half in slot nr for execution. Typically, an   
 interrupt handler will mark its bottom half (hence the name!) for execution at a   
 "safer time".  
   
Bottom halves are globally locked tasklets, so the question "when are bottom half   
 handlers executed?" is really "when are tasklets executed?". And the answer is, in   
 two places: a) on each schedule() and b) on each interrupt/syscall return path in   
 entry.S (TODO: therefore, the schedule() case is really boring - it like adding yet  
 another very very slow interrupt, why not get rid of handle\_softirq label from   
 schedule() altogether?).  
   
2.8 Task Queues  
   
Task queues can be though of as a dynamic extension to old bottom halves. In fact, in  
 the source code they are sometimes referred to as "new" bottom halves. More   
 specifically, the old bottom halves discussed in previous section have these   
 limitations:  
   
1. There are only a fixed number (32) of them.  
 2. Each bottom half can only be associated with one handler function.  
 3. Bottom halves are consumed with a spinlock held so they cannot block.  
   
So, with task queues, arbitrary number of functions can be chained and processed one   
 after another at a later time. One creates a new task queue using the   
 DECLARE\_TASK\_QUEUE() macro and queues a task onto it using the queue\_task()   
 function. The task queue then can be processed using run\_task\_queue(). Instead of   
 creating your own task queue (and having to consume it manually) you can use one of  
 Linux' predefined task queues which are consumed at well-known points:  
   
1. tq\_timer: the timer task queue, run on each timer interrupt and when releasing a   
 tty device (closing or releasing a half-opened terminal device). Since the timer   
 handler runs in interrupt context, the tq\_timer tasks also run in interrupt context  
 and thus cannot block.  
   
2. tq\_scheduler: the scheduler task queue, consumed by the scheduler (and also when   
 closing tty devices, like tq\_timer). Since the scheduler executed in the context of  
 the process being re-scheduled, the tq\_scheduler tasks can do anything they like,   
 i.e. block, use process context data (but why would they want to), etc.  
   
3. tq\_immediate: this is really a bottom half IMMEDIATE\_BH, so drivers can   
 queue\_task(task, &tq\_immediate) and then mark\_bh(IMMEDIATE\_BH) to be consumed in   
 interrupt context.  
   
4. tq\_disk: used by low level block device access (and RAID) to start the actual   
 requests. This task queue is exported to modules but shouldn't be used except for   
 the special purposes which it was designed for.  
   
Unless a driver uses its own task queues, it does not need to call run\_tasks\_queues()  
 to process the queue, except under circumstances explained below.  
   
The reason tq\_timer/tq\_scheduler task queues are consumed not only in the usual   
 places but elsewhere (closing tty device is but one example) becomes clear if one   
 remembers that the driver can schedule tasks on the queue, and these tasks only   
 make sense while a particular instance of the device is still valid - which usually  
 means until the application closes it. So, the driver may need to call   
 run\_task\_queue() to flush the tasks it (and anyone else) has put on the queue,   
 because allowing them to run at a later time may make no sense - i.e. the relevant   
 data structures may have been freed/reused by a different instance. This is the   
 reason you see run\_task\_queue() on tq\_timer and tq\_scheduler in places other than   
 timer interrupt and schedule() respectively.  
   
2.9 Tasklets  
   
Not yet, will be in future revision.  
   
2.10 Softirqs  
   
Not yet, will be in future revision.  
   
2.11 How System Calls Are Implemented on i386 Architecture?  
   
There are two mechanisms under Linux for implementing system calls:  
   
\* lcall7/lcall27 call gates;  
 \* int 0x80 software interrupt.  
   
Native Linux programs use int 0x80 whilst binaries from foreign flavours of UNIX   
 (Solaris, UnixWare 7 etc.) use the lcall7 mechanism. The name 'lcall7' is   
 historically misleading because it also covers lcall27 (e.g. Solaris/x86), but the   
 handler function is called lcall7\_func.  
   
When the system boots, the function arch/i386/kernel/traps.c:trap\_init() is called   
 which sets up the IDT so that vector 0x80 (of type 15, dpl 3) points to the address  
 of system\_call entry from arch/i386/kernel/entry.S.  
   
When a userspace application makes a system call, the arguments are passed via   
 registers and the application executes 'int 0x80' instruction. This causes a trap   
 into kernel mode and processor jumps to system\_call entry point in entry.S. What   
 this does is:  
   
1. Save registers.  
   
2. Set %ds and %es to KERNEL\_DS, so that all data (and extra segment) references are   
 made in kernel address space.  
   
3. If the value of %eax is greater than NR\_syscalls (currently 256), fail with ENOSYS  
 error.  
   
4. If the task is being ptraced (tsk->ptrace & PF\_TRACESYS), do special processing.   
 This is to support programs like strace (analogue of SVR4 truss(1)) or debuggers.  
   
5. Call sys\_call\_table+4\*(syscall\_number from %eax). This table is initialised in the  
 same file (arch/i386/kernel/entry.S) to point to individual system call handlers   
 which under Linux are (usually) prefixed with sys\_, e.g. sys\_open, sys\_exit, etc.   
 These C system call handlers will find their arguments on the stack where SAVE\_ALL   
 stored them.  
   
6. Enter 'system call return path'. This is a separate label because it is used not   
 only by int 0x80 but also by lcall7, lcall27. This is concerned with handling   
 tasklets (including bottom halves), checking if a schedule() is needed (tsk->  
 need\_resched != 0), checking if there are signals pending and if so handling them.  
   
Linux supports up to 6 arguments for system calls. They are passed in %ebx, %ecx,   
 %edx, %esi, %edi (and %ebp used temporarily, see \_syscall6() in asm-i386/unistd.h).  
 The system call number is passed via %eax.  
   
2.12 Atomic Operations  
   
There are two types of atomic operations: bitmaps and atomic\_t. Bitmaps are very   
 convenient for maintaining a concept of "allocated" or "free" units from some large  
 collection where each unit is identified by some number, for example free inodes or  
 free blocks. They are also widely used for simple locking, for example to provide   
 exclusive access to open a device. An example of this can be found in   
 arch/i386/kernel/microcode.c:  
   
/\*  
 \* Bits in microcode\_status. (31 bits of room for future expansion)  
 \*/  
 #define MICROCODE\_IS\_OPEN 0 /\* set if device is in use \*/  
   
static unsigned long microcode\_status;  
   
There is no need to initialise microcode\_status to 0 as BSS is zero-cleared under   
 Linux explicitly.  
   
/\*  
 \* We enforce only one user at a time here with open/close.  
 \*/  
 static int microcode\_open(struct inode \*inode, struct file \*file)  
 {  
 if (!capable(CAP\_SYS\_RAWIO))  
 return -EPERM;  
   
/\* one at a time, please \*/  
 if (test\_and\_set\_bit(MICROCODE\_IS\_OPEN, &microcode\_status))  
 return -EBUSY;  
   
MOD\_INC\_USE\_COUNT;  
 return 0;  
 }  
   
The operations on bitmaps are:  
   
\* void set\_bit(int nr, volatile void \*addr): set bit nr in the bitmap pointed to by   
 addr.  
 \* void clear\_bit(int nr, volatile void \*addr): clear bit nr in the bitmap pointed to   
 by addr.  
 \* void change\_bit(int nr, volatile void \*addr): toggle bit nr (if set clear, if clear  
 set) in the bitmap pointed to by addr.  
 \* int test\_and\_set\_bit(int nr, volatile void \*addr): atomically set bit nr and return  
 the old bit value.  
 \* int test\_and\_clear\_bit(int nr, volatile void \*addr): atomically clear bit nr and   
 return the old bit value.  
 \* int test\_and\_change\_bit(int nr, volatile void \*addr): atomically toggle bit nr and   
 return the old bit value.  
   
These operations use the LOCK\_PREFIX macro, which on SMP kernels evaluates to bus   
 lock instruction prefix and to nothing on UP. This guarantees atomicity of access   
 in SMP environment.  
   
Sometimes bit manipulations are not convenient, but instead we need to perform   
 arithmetic operations - add, subtract, increment decrement. The typical cases are   
 reference counts (e.g. for inodes). This facility is provided by the atomic\_t data   
 type and the following operations:  
   
\* atomic\_read(&v): read the value of atomic\_t variable v.  
 \* atomic\_set(&v, i): set the value of atomic\_t variable v to integer i.  
 \* void atomic\_add(int i, volatile atomic\_t \*v): add integer i to the value of atomic   
 variable pointed to by v.  
 \* void atomic\_sub(int i, volatile atomic\_t \*v): subtract integer i from the value of   
 atomic variable pointed to by v.  
 \* int atomic\_sub\_and\_test(int i, volatile atomic\_t \*v): subtract integer i from the   
 value of atomic variable pointed to by v; return 1 if the new value is 0, return 0   
 otherwise.  
 \* void atomic\_inc(volatile atomic\_t \*v): increment the value by 1.  
 \* void atomic\_dec(volatile atomic\_t \*v): decrement the value by 1.  
 \* int atomic\_dec\_and\_test(volatile atomic\_t \*v): decrement the value; return 1 if the  
 new value is 0, return 0 otherwise.  
 \* int atomic\_inc\_and\_test(volatile atomic\_t \*v): increment the value; return 1 if the  
 new value is 0, return 0 otherwise.  
 \* int atomic\_add\_negative(int i, volatile atomic\_t \*v): add the value of i to v and   
 return 1 if the result is negative. Return 0 if the result is greater than or equal  
 to 0. This operation is used for implementing semaphores.  
   
2.13 Spinlocks, Read-write Spinlocks and Big-Reader Spinlocks  
   
Since the early days of Linux support (early 90s, this century), developers were   
 faced with the classical problem of accessing shared data between different types   
 of context (user process vs interrupt) and different instances of the same context   
 from multiple cpus.  
   
SMP support was added to Linux 1.3.42 on 15 Nov 1995 (the original patch was made to   
 1.3.37 in October the same year).  
   
If the critical region of code may be executed by either process context and   
 interrupt context, then the way to protect it using cli/sti instructions on UP is:  
   
unsigned long flags;  
   
save\_flags(flags);  
 cli();  
 /\* critical code \*/  
 restore\_flags(flags);  
   
While this is ok on UP, it obviously is of no use on SMP because the same code   
 sequence may be executed simultaneously on another cpu, and while cli() provides   
 protection against races with interrupt context on each CPU individually, it   
 provides no protection at all against races between contexts running on different   
 CPUs. This is where spinlocks are useful for.  
   
There are three types of spinlocks: vanilla (basic), read-write and big-reader   
 spinlocks. Read-write spinlocks should be used when there is a natural tendency of   
 'many readers and few writers'. Example of this is access to the list of registered  
 filesystems (see fs/super.c). The list is guarded by the file\_systems\_lock   
 read-write spinlock because one needs exclusive access only when   
 registering/unregistering a filesystem, but any process can read the file   
 /proc/filesystems or use the sysfs(2) system call to force a read-only scan of the   
 file\_systems list. This makes it sensible to use read-write spinlocks. With   
 read-write spinlocks, one can have multiple readers at a time but only one writer   
 and there can be no readers while there is a writer.   
  
Btw, it would be nice if new   
 readers would not get a lock while there is a writer trying to get a lock, i.e. if   
 Linux could correctly deal with the issue of potential writer starvation by   
 multiple readers. This would mean that readers must be blocked while there is a   
 writer attempting to get the lock. This is not currently the case and it is not   
 obvious whether this should be fixed - the argument to the contrary is - readers   
 usually take the lock for a very short time so should they really be starved while   
 the writer takes the lock for potentially longer periods?  
   
Big-reader spinlocks are a form of read-write spinlocks heavily optimised for very   
 light read access, with a penalty for writes. There is a limited number of   
 big-reader spinlocks - currently only two exist, of which one is used only on   
 sparc64 (global irq) and the other is used for networking. In all other cases where  
 the access pattern does not fit into any of these two scenarios, one should use   
 basic spinlocks. You cannot block while holding any kind of spinlock.  
   
Spinlocks come in three flavours: plain, \_irq() and \_bh().  
   
1. Plain spin\_lock()/spin\_unlock(): if you know the interrupts are always disabled or  
 if you do not race with interrupt context (e.g. from within interrupt handler),   
 then you can use this one. It does not touch interrupt state on the current CPU.  
   
2. spin\_lock\_irq()/spin\_unlock\_irq(): if you know that interrupts are always enabled   
 then you can use this version, which simply disables (on lock) and re-enables (on   
 unlock) interrupts on the current CPU. For example, rtc\_read() uses spin\_lock\_irq(&  
 rtc\_lock) (interrupts are always enabled inside read()) whilst rtc\_interrupt() uses  
 spin\_lock(&rtc\_lock) (interrupts are always disabled inside interrupt handler).   
 Note that rtc\_read() uses spin\_lock\_irq() and not the more generic   
 spin\_lock\_irqsave() because on entry to any system call interrupts are always   
 enabled.  
   
3. spin\_lock\_irqsave()/spin\_unlock\_irqrestore(): the strongest form, to be used when   
 the interrupt state is not known, but only if interrupts matter at all, i.e. there   
 is no point in using it if our interrupt handlers don't execute any critical code.  
   
The reason you cannot use plain spin\_lock() if you race against interrupt handlers is  
 because if you take it and then an interrupt comes in on the same CPU, it will busy  
 wait for the lock forever: the lock holder, having been interrupted, will not   
 continue until the interrupt handler returns.  
   
The most common usage of a spinlock is to access a data structure shared between user  
 process context and interrupt handlers:  
   
spinlock\_t my\_lock = SPIN\_LOCK\_UNLOCKED;  
   
my\_ioctl()  
 {  
 spin\_lock\_irq(&my\_lock);  
 /\* critical section \*/  
 spin\_unlock\_irq(&my\_lock);  
 }  
   
my\_irq\_handler()  
 {  
 spin\_lock(&lock);  
 /\* critical section \*/  
 spin\_unlock(&lock);  
 }  
   
There are a couple of things to note about this example:  
   
1. The process context, represented here as a typical driver method - ioctl()   
 (arguments and return values omitted for clarity), must use spin\_lock\_irq() because  
 it knows that interrupts are always enabled while executing the device ioctl()   
 method.  
 2. Interrupt context, represented here by my\_irq\_handler() (again arguments omitted   
 for clarity) can use plain spin\_lock() form because interrupts are disabled inside   
 an interrupt handler.  
   
2.14 Semaphores and read/write Semaphores  
   
Sometimes, while accessing a shared data structure, one must perform operations that   
 can block, for example copy data to userspace. The locking primitive available for   
 such scenarios under Linux is called a semaphore. There are two types of   
 semaphores: basic and read-write semaphores. Depending on the initial value of the   
 semaphore, they can be used for either mutual exclusion (initial value of 1) or to   
 provide more sophisticated type of access.  
   
Read-write semaphores differ from basic semaphores in the same way as read-write   
 spinlocks differ from basic spinlocks: one can have multiple readers at a time but   
 only one writer and there can be no readers while there are writers - i.e. the   
 writer blocks all readers and new readers block while a writer is waiting.  
   
Also, basic semaphores can be interruptible - just use the operations   
 down/up\_interruptible() instead of the plain down()/up() and check the value   
 returned from down\_interruptible(): it will be non zero if the operation was   
 interrupted.  
   
Using semaphores for mutual exclusion is ideal in situations where a critical code   
 section may call by reference unknown functions registered by other   
 subsystems/modules, i.e. the caller cannot know apriori whether the function blocks  
 or not.  
   
A simple example of semaphore usage is in kernel/sys.c, implementation of   
 gethostname(2)/sethostname(2) system calls.  
   
asmlinkage long sys\_sethostname(char \*name, int len)  
 {  
 int errno;  
   
if (!capable(CAP\_SYS\_ADMIN))  
 return -EPERM;  
 if (len < 0 || len > \_\_NEW\_UTS\_LEN)  
 return -EINVAL;  
 down\_write(&uts\_sem);  
 errno = -EFAULT;  
 if (!copy\_from\_user(system\_utsname.nodename, name, len)) {  
 system\_utsname.nodename[len] = 0;  
 errno = 0;  
 }  
 up\_write(&uts\_sem);  
 return errno;  
 }  
   
asmlinkage long sys\_gethostname(char \*name, int len)  
 {  
 int i, errno;  
   
if (len < 0)  
 return -EINVAL;  
 down\_read(&uts\_sem);  
 i = 1 + strlen(system\_utsname.nodename);  
 if (i > len)  
 i = len;  
 errno = 0;  
 if (copy\_to\_user(name, system\_utsname.nodename, i))  
 errno = -EFAULT;  
 up\_read(&uts\_sem);  
 return errno;  
 }  
   
The points to note about this example are:  
   
1. The functions may block while copying data from/to userspace in   
 copy\_from\_user()/copy\_to\_user(). Therefore they could not use any form of spinlock   
 here.  
 2. The semaphore type chosen is read-write as opposed to basic because there may be   
 lots of concurrent gethostname(2) requests which need not be mutually exclusive.  
   
Although Linux implementation of semaphores and read-write semaphores is very   
 sophisticated, there are possible scenarios one can think of which are not yet   
 implemented, for example there is no concept of interruptible read-write   
 semaphores. This is obviously because there are no real-world situations which   
 require these exotic flavours of the primitives.  
   
2.15 Kernel Support for Loading Modules  
   
Linux is a monolithic operating system and despite all the modern hype about some   
 "advantages" offered by operating systems based on micro-kernel design, the truth   
 remains (quoting Linus Torvalds himself):  
   
... message passing as the fundamental operation of the OS is just an exercise in   
 computer science masturbation. It may feel good, but you don't actually get   
 anything DONE.  
   
Therefore, Linux is and will always be based on a monolithic design, which means that  
 all subsystems run in the same privileged mode and share the same address space;   
 communication between them is achieved by the usual C function call means.  
   
However, although separating kernel functionality into separate "processes" as is   
 done in micro-kernels is definitely a bad idea, separating it into dynamically   
 loadable on demand kernel modules is desirable in some circumstances (e.g. on   
 machines with low memory or for installation kernels which could otherwise contain   
 ISA auto-probing device drivers that are mutually exclusive). The decision whether   
 to include support for loadable modules is made at compile time and is determined   
 by the CONFIG\_MODULES option. Support for module autoloading via request\_module()   
 mechanism is a separate compilation option (CONFIG\_KMOD).  
   
The following functionality can be implemented as loadable modules under Linux:  
   
1. Character and block device drivers, including misc device drivers.  
 2. Terminal line disciplines.  
 3. Virtual (regular) files in /proc and in devfs (e.g. /dev/cpu/microcode vs   
 /dev/misc/microcode).  
 4. Binary file formats (e.g. ELF, aout, etc).  
 5. Execution domains (e.g. Linux, UnixWare7, Solaris, etc).  
 6. Filesystems.  
 7. System V IPC.  
   
There a few things that cannot be implemented as modules under Linux (probably   
 because it makes no sense for them to be modularised):  
   
1. Scheduling algorithms.  
 2. VM policies.  
 3. Buffer cache, page cache and other caches.  
   
Linux provides several system calls to assist in loading modules:  
   
1. caddr\_t create\_module(const char \*name, size\_t size): allocates size bytes using   
 vmalloc() and maps a module structure at the beginning thereof. This new module is   
 then linked into the list headed by module\_list. Only a process with CAP\_SYS\_MODULE  
 can invoke this system call, others will get EPERM returned.  
   
2. long init\_module(const char \*name, struct module \*image): loads the relocated   
 module image and causes the module's initialisation routine to be invoked. Only a   
 process with CAP\_SYS\_MODULE can invoke this system call, others will get EPERM   
 returned.  
   
3. long delete\_module(const char \*name): attempts to unload the module. If name ==   
 NULL, attempt is made to unload all unused modules.  
   
4. long query\_module(const char \*name, int which, void \*buf, size\_t bufsize, size\_t   
 \*ret): returns information about a module (or about all modules).  
   
The command interface available to users consists of:  
   
\* insmod: insert a single module.  
 \* modprobe: insert a module including all other modules it depends on.  
 \* rmmod: remove a module.  
 \* modinfo: print some information about a module, e.g. author, description,   
 parameters the module accepts, etc.  
   
Apart from being able to load a module manually using either insmod or modprobe, it   
 is also possible to have the module inserted automatically by the kernel when a   
 particular functionality is required. The kernel interface for this is the function  
 called request\_module(name) which is exported to modules, so that modules can load   
 other modules as well. The request\_module(name) internally creates a kernel thread   
 which execs the userspace command modprobe -s -k module\_name, using the standard   
 exec\_usermodehelper() kernel interface (which is also exported to modules). The   
 function returns 0 on success, however it is usually not worth checking the return   
 code from request\_module(). Instead, the programming idiom is:  
   
if (check\_some\_feature() == NULL)  
 request\_module(module);  
 if (check\_some\_feature() == NULL)  
 return -ENODEV;  
   
For example, this is done by fs/block\_dev.c:get\_blkfops() to load a module   
 block-major-N when attempt is made to open a block device with major N. Obviously,   
 there is no such module called block-major-N (Linux developers only chose sensible   
 names for their modules) but it is mapped to a proper module name using the file   
 /etc/modules.conf. However, for most well-known major numbers (and other kinds of   
 modules) the modprobe/insmod commands know which real module to load without   
 needing an explicit alias statement in /etc/modules.conf.  
   
A good example of loading a module is inside the mount(2) system call. The mount(2)   
 system call accepts the filesystem type as a string which fs/super.c:do\_mount()   
 then passes on to fs/super.c:get\_fs\_type():  
   
static struct file\_system\_type \*get\_fs\_type(const char \*name)  
 {  
 struct file\_system\_type \*fs;  
   
read\_lock(&file\_systems\_lock);  
 fs = \*(find\_filesystem(name));  
 if (fs && !try\_inc\_mod\_count(fs->owner))  
 fs = NULL;  
 read\_unlock(&file\_systems\_lock);  
 if (!fs && (request\_module(name) == 0)) {  
 read\_lock(&file\_systems\_lock);  
 fs = \*(find\_filesystem(name));  
 if (fs && !try\_inc\_mod\_count(fs->owner))  
 fs = NULL;  
 read\_unlock(&file\_systems\_lock);  
 }  
 return fs;  
 }  
   
A few things to note in this function:  
   
1. First we attempt to find the filesystem with the given name amongst those already   
 registered. This is done under protection of file\_systems\_lock taken for read (as   
 we are not modifying the list of registered filesystems).  
   
2. If such a filesystem is found then we attempt to get a new reference to it by   
 trying to increment its module's hold count. This always returns 1 for statically   
 linked filesystems or for modules not presently being deleted. If   
 try\_inc\_mod\_count() returned 0 then we consider it a failure - i.e. if the module   
 is there but is being deleted, it is as good as if it were not there at all.  
   
3. We drop the file\_systems\_lock because what we are about to do next   
 (request\_module()) is a blocking operation, and therefore we can't hold a spinlock   
 over it. Actually, in this specific case, we would have to drop file\_systems\_lock   
 anyway, even if request\_module() were guaranteed to be non-blocking and the module   
 loading were executed in the same context atomically. The reason for this is that   
 the module's initialisation function will try to call register\_filesystem(), which   
 will take the same file\_systems\_lock read-write spinlock for write.  
   
4. If the attempt to load was successful, then we take the file\_systems\_lock spinlock  
 and try to locate the newly registered filesystem in the list. Note that this is   
 slightly wrong because it is in principle possible for a bug in modprobe command to  
 cause it to coredump after it successfully loaded the requested module, in which   
 case request\_module() will fail even though the new filesystem will be registered,   
 and yet get\_fs\_type() won't find it.  
   
5. If the filesystem is found and we are able to get a reference to it, we return it.  
 Otherwise we return NULL.  
   
When a module is loaded into the kernel, it can refer to any symbols that are   
 exported as public by the kernel using EXPORT\_SYMBOL() macro or by other currently   
 loaded modules. If the module uses symbols from another module, it is marked as   
 depending on that module during dependency recalculation, achieved by running   
 depmod -a command on boot (e.g. after installing a new kernel).  
   
Usually, one must match the set of modules with the version of the kernel interfaces   
 they use, which under Linux simply means the "kernel version" as there is no   
 special kernel interface versioning mechanism in general. However, there is a   
 limited functionality called "module versioning" or CONFIG\_MODVERSIONS which allows  
 to avoid recompiling modules when switching to a new kernel. What happens here is   
 that the kernel symbol table is treated differently for internal access and for   
 access from modules. The elements of public (i.e. exported) part of the symbol   
 table are built by 32bit checksumming the C declaration.   
  
So, in order to resolve a   
 symbol used by a module during loading, the loader must match the full   
 representation of the symbol that includes the checksum; it will refuse to load the  
 module if these symbols differ. This only happens when both the kernel and the   
 module are compiled with module versioning enabled. If either one of them uses the   
 original symbol names, the loader simply tries to match the kernel version declared  
 by the module and the one exported by the kernel and refuses to load if they   
 differ.  
  
======================================  
 Paging and Swapping Linux:  
======================================  
   
Swapping  
 ========  
 Some systems are pure swapping systems, some systems are pure paging systems and   
 others are mixed mode systems.  
   
Originally Unix system V was a pure swapping system.  
   
To swap a process means to move that entire process out of main memory and to the   
 swap area on hard disk, whereby all pages of that process are moved at the same   
 time.  
   
This carried the disadvantage of a performance penalty. When a swapped out process   
 becomes active and moves from the sleep queue to the run queue, the kernel has to   
 load an entire process (perhaps many pages of memory) back into RAM from the swap   
 space. With large processes this is understandably slow. Enter paging.  
   
Paging  
 ======  
 Paging was introduced as a solution to the inefficiency of swapping entire processes   
 in and out of memory at once.  
   
With paging, when the kernel requires more main memory for an active process, only   
 the least recently used pages of processes are moved to the swap space.  
   
Therefore when a process that has paged out memory becomes active, it is likely that   
 it will not need access to the pages of memory that have been paged out to the swap  
 space, and if it does then at least only a few pages need to be transferred between  
 disk and RAM.  
   
Paging was first implemented in system V[?] in 19??  
 The working sets  
   
For efficient paging, the kernel needs to keep regular statistics on the memory   
 activity of processes it keeps track of which pages a process has most recently   
 used. These pages are known as the working set.  
   
When the kernel needs memory, it will prefer to keep pages in the working sets of   
 processes in RAM as long as possible and to rather page out the other less recently  
 used pages as they have statistically been proven to be less frequently accessed,   
 and therefore unlikely to be accesses again in the near future.  
 Implementation of swapping and paging in different systems  
   
Current Unix systems use the following methods of memory management:  
   
\* SVR3 and newer based systems are mixed swapping and paging systems, as is FreeBSD.   
 Paging is normally used but if memory usage runs extremely heavy, too quickly for   
 the kernels' pager to page out enough pages of memory, then the system will revert   
 to swapping. This technique is also known as desperation swapping.  
   
\* Linux is a pure paging system it never swaps, neither under normal usage nor does   
 it employ desperation swapping under heavy usage.  
   
\* When the FreeBSD VM system is critically low on RAM or swap, it will lock the   
 largest process, and then flush all dirty vnode-backed pages - and will move active  
 pages into the inactive queue, allowing them to be reclaimed. If, after all of   
 that, there is still not enough memory available for the locked process, only then   
 will the process be killed.  
   
\* Under emergency memory situations when Linux runs out of memory (both physical and   
 swap combined) the kernel starts killing processes. It uses an algorithm to work   
 out which process to kill first - it tries to kill offending memory hogs that have   
 been running for a short amount of time first before less used processes that have   
 been running for a long time, which are most likely important system services. This  
 functionality is known as the out of memory (OOM) killer.[5]  
   
Virtual memory  
 ==============  
 Virtual memory can mean two different things, in different contexts. Firstly it can   
 refer to only swap memory; secondly it could refer to the combination of both RAM   
 and swap memory.  
  
======================================  
 DEBUGGING ////////////////////////  
======================================  
  
======================================  
 Debugging Solaris:  
======================================  
   
cat -v -t -e [file]  
 /\* Show non-printing characters \*/  
=======================================  
   
dumpadm -d swap  
 /\* Configure swap device as dump device \*/  
=======================================  
   
ld -l <libname without 'lib'>  
 /\* Check if you have a particular library \*/  
=======================================  
   
truss -f -p <pid of a shell>  
 /\* Using multiple windows, this can be used to trace setuid/setgid programs \*/  
=======================================  
   
truss executable  
 /\* Trace doing of given command ( useful debugging ) \*/  
=======================================  
  
======================================  
 Web Services :  
======================================  
  
 Definitions:   
   
XML: Short for Extensible Markup Language, a specification developed by the W3C. XML   
 is a pared-down version of SGML, designed especially for Web documents. It allows   
 designers to create their own customized tags, enabling the definition,   
 transmission, validation, and interpretation of data between applications and   
 between organizations.   
   
SOAP: Short for Simple Object Access Protocol, a lightweight XML-based messaging   
 protocol used to encode the information in Web service request and response   
 messages before sending them over a network. SOAP messages are independent of any   
 operating system or protocol and may be transported using a variety of Internet   
 protocols, including SMTP, MIME, and HTTP.   
   
WSDL: Short for Web Services Description Language, an XML-formatted language used to   
 describe a Web service's capabilities as collections of communication endpoints   
 capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based   
 worldwide business registry. WSDL is the language that UDDI uses. WSDL was   
 developed jointly by Microsoft and IBM.   
   
UDDI: Short for Universal Description, Discovery and Integration. A Web-based   
 distributed directory that enables businesses to list themselves on the Internet   
 and discover each other, similar to a traditional phone book's yellow and white   
 pages.   
   
The term Web services describes a standardized way of integrating Web-based   
 applications using the XML, SOAP, WSDL and UDDI open standards over an Internet   
 protocol backbone. XML is used to tag the data, SOAP is used to transfer the data,   
 WSDL is used for describing the services available and UDDI is used for listing   
 what services are available. Used primarily as a means for businesses to   
 communicate with each other and with clients, Web services allow organizations to   
 communicate data without intimate knowledge of each other's IT systems behind the   
 firewall.   
   
Unlike traditional client/server models, such as a Web server/Web page system, Web   
 services do not provide the user with a GUI. Web services instead share business   
 logic, data and processes through a programmatic interface across a network. The   
 applications interface, not the users. Developers can then add the Web service to a  
 GUI (such as a Web page or an executable program) to offer specific functionality   
 to users.   
   
Web services allow different applications from different sources to communicate with   
 each other without time-consuming custom coding, and because all communication is   
 in XML, Web services are not tied to any one operating system or programming   
 language. For example, Java can talk with Perl, Windows applications can talk with   
 UNIX applications.   
   
Web services do not require the use of browsers or HTML.   
   
Web services are sometimes called application services.   
   
Web Services

   
   
 ======================================   
 Systems Monitoring:   
 ======================================   
   
 A typical systems management tool is a collection of software programs that work in   
 unison like Nagios which uses SNMP to query your switches and routers and it uses   
 NSClinet++ to monitor your Windows hosts. It reports back to you what's up and   
 what's down on the network and what the status is of each device. Here are some   
 screen shots of my setup.   
   
 Nagios Alerts   
   
   
   
 Nagios Host Groups   
   
   
   
 Nagios Hosts   
   
   
   
   
 Nagios Map   
   
   
   
   
 Nagios Scheduling   
   
   
   
 Nagios Services   
   
   
   
   
 Nagios Summary   
   
   
   
   
As you can see I need to fix monitoring of port 1 on the router. Hahaha.Smile   
Other tools out there would be [Foglight](http://www.quest.com/application-monitoring-capabilities/infrastructure-management.aspx), [Netcool](http://www-01.ibm.com/software/tivoli/solutions/server-network-device/), [HP Openview](https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10%5E36657_4000_100__), and  [Big Brother](http://bb4.com/) just to name a few.

======================================  
 JMS Messaging :  
======================================

What Is the Java Message Service?   
   
An enterprise messaging system, also referred to as message-oriented middleware   
 (MOM), enables applications to communicate with one another through the exchange of  
 messages. A message is a request, report, and/or event that contains information   
 needed to coordinate communication between different applications. A message   
 provides a level of abstraction, allowing you to separate the details about the   
 destination system from the application code.   
   
The Java Message Service (JMS) is a standard API for accessing enterprise messaging   
 systems. Specifically, JMS:   
   
 \* Enables Java applications sharing a messaging system to exchange messages   
 \* Simplifies application development by providing a standard interface for   
 creating, sending, and receiving messages   
   
The following figure illustrates WebLogic JMS messaging.   
   
Figure 1-1 WebLogic JMS Messaging   
   
WebLogic JMS Messaging

======================================  
Java Server Pages (JSP) :  
======================================

JavaServer Pages (JSP) is a Java technology that allows software developers to create  
 dynamically-generated web sites, with HTML, XML, or other document types, in   
 response to a Web client request. The technology allows Java code and certain   
 pre-defined actions to be embedded into static content.   
   
The JSP syntax adds additional XML-like tags, called JSP actions, to be used to   
 invoke built-in functionality. Additionally, the technology allows for the creation  
 of JSP tag libraries that act as extensions to the standard HTML or XML tags. Tag   
 libraries provide a platform independent way of extending the capabilities of a Web  
 server.   
   
JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a  
 servlet in Java code that is then compiled by the Java compiler, or it may generate  
 byte code for the servlet directly. JSPs can also be interpreted on-the-fly,   
 reducing the time taken to reload changes.

======================================  
 Database Triggers :  
 ======================================

A database trigger is procedural code that is automatically executed in response to   
 certain events on a particular table in a database. Triggers can restrict access to  
 specific data, perform logging, or audit data modifications.   
   
There are two classes of triggers, they are either "row triggers" or "statement   
 triggers". Row triggers define an action for every row of a table, while statement   
 triggers occur only once per INSERT, UPDATE, or DELETE statement. Triggers cannot   
 be used to audit data retrieval via SELECT statements.   
   
Each class can be of several types. There are "BEFORE triggers" and "AFTER triggers"   
 which identifies the time of execution of the trigger. There is also an "INSTEAD OF  
 trigger" which is a trigger that will execute instead of the triggering statement.   
   
There are typically three triggering events that cause triggers to 'fire':   
   
 \* INSERT event (as a new record is being inserted into the database).   
 \* UPDATE event (as a record is being changed).   
 \* DELETE event (as a record is being deleted).   
   
The trigger is used to automate DML condition process.   
   
The major features of database triggers, and their effects, are:   
   
 \* do not accept parameters or arguments (but may store affected-data in temporary  
 tables)   
 \* cannot perform commit or rollback operations because they are part of the   
 triggering SQL statement (only through autonomous transactions)   
 \* can cause mutating table errors, if they are poorly written.   
   
In addition to triggers that fire when data is modified, Oracle 9i supports triggers   
 that fire when schema objects (that is, tables) are modified and when user logon or  
 logoff events occur. These trigger types are referred to as "Schema-level   
 triggers".   
   
Schema-level triggers   
   
 \* After Creation   
 \* Before Alter   
 \* After Alter   
 \* Before Drop   
 \* After Drop   
 \* Before Logoff   
 \* After Logon   
   
The two main types of triggers are:   
   
1) Row Level Trigger 2) Statement Level Trigger   
   
Based on the 2 types of classifications, we could have 12 types of triggers.   
   
 \* Before Insert row level   
 \* After Insert row level   
 \* Before Delete row level   
 \* After Delete row level   
 \* Before Update row level   
 \* After Update row level   
 \* Before Insert Statement Level   
 \* After Insert Statement Level   
 \* Before Delete Statement Level   
 \* After Delete Statement Level   
 \* Before Update Statement Level   
 \* After Update Statement Level   
   
   
   
MySQL 5.0.2 introduced support for triggers. Some of the triggers MYSQL supports are   
   
 \* INSERT Trigger   
 \* UPDATE Trigger   
 \* DELETE Trigger   
   
The SQL:2003 standard mandates that triggers give programmers access to record   
 variables by means of a syntax such as REFERENCING NEW AS n. For example, if a   
 trigger is monitoring for changes to a salary column one could write a trigger like  
 the following:   
   
CREATE TRIGGER salary\_trigger   
BEFORE UPDATE ON employee\_table   
REFERENCING NEW ROW AS n, OLD ROW AS o   
FOR EACH ROW   
IF n.salary <> o.salary THEN   
   
END IF;  
  
======================================  
 Compile Linux Kernel :  
======================================  
   
   
 1. How to Compile 2.6 kernel for RedHat 9/8   
 Mike Chirico   
 Last Updated: Wed Mar 24 09:12:06 EST 2004   
   
 The latest version of this document can be found at:   
 http://prdownloads.sourceforge.net/souptonuts/README\_26.txt?download   
   
 For configs ref:   
 http://sourceforge.net/project/showfiles.php?group\_id=79320&package\_id=109944   
   
   
   
 STEP 1:   
   
 Download the latest version of the kernel and any patches.   
 This documentation is done with linux-2.6.3, but look for   
 later versions.   
 http://www.kernel.org/pub/linux/kernel/v2.6/   
   
 Also take a look at   
 http://www.codemonkey.org.uk/post-halloween-2.5.txt This has   
 some useful hints on some of the changes needed.   
   
   
 STEP 2:   
   
 Download the latest version of module-init-tools   
 "module-init-tools-3.0.tar.gz" and   
 "modutils-2.4.21-23.src.rpm"   
   
 http://www.kernel.org/pub/linux/kernel/people/rusty/modules/module-init-tools-3  
 .0.tar.gz   
 http://www.kernel.org/pub/linux/kernel/people/rusty/modules/modutils-2.4.21-23.  
 src.rpm   
  
   
 STEP 3:   
   
 Install module-init-tools. This will replace depmod   
 [/sbin/depmod] and other tools.   
   
 tar -zxvf module-init-tools-3.0.tar.gz   
 cd module-init-tools-3.0   
 ./configure --prefix=/sbin   
 make   
 make install   
 ./generate-modprobe.conf /etc/modprobe.conf   
   
  
 STEP 4:   
   
 Install modutils-2.4.21-23.src.rpm. You may get warnings   
 about user rusty and group rusty not existing. Also, yes,   
 you'll have to force the install. If you don't do these steps   
 for both Redhat 9 and Redhat 8, you'll have problems with the   
 make modules\_install.   
   
 rpm -i modutils-2.4.21-23.src.rpm   
 rpmbuild -bb /usr/src/redhat/SPECS/modutils.spec   
 rpm -Fi /usr/src/redhat/RPMS/i386/modutils-2.4.21-23.i386.rpm   
   
   
 STEP 5:   
   
 Install and configure the kernel. Do NOT use the /usr/src/linux   
 area! Reference the README. I put my files in /home/src/kernel/   
   
 gunzip linux-2.6.3.tar.gz tar -xvf linux-2.6.3.tar cd   
   
 linux-2.6.3   
   
 If you have patches install these now:   
   
 bzip2 -dc ../patch-2.6.xx.bz2 | patch -p1   
   
   
 STEP 6:   
   
 Copy the appropriate /usr/src/linux-2.4/configs   
 [kernel-2.4.20-i686.config, kernel-2.4.20-i686-smp.config]   
 to .config in whatever directory you are installing. In my   
 case it's /home/src/kernel/linux-2.6.3   
   
 cp /usr/src/linux-2.4/configs/kernel-2.4.20-i686.config \   
 /home/src/kernel/linux-2.6.3/.config   
   
 If you don't have the source configs, you can download them   
 from here:   
   
 https://sourceforge.net/project/showfiles.php?group\_id=79320&package\_id=109944   
   
 I've also included a file config2.6-chirico which was a 2.6   
 version for some of my systems . This isn't a bad reference if   
 you run into trouble.   
  
   
 STEP 7:   
   
 Assuming you copied the appropriate kernel-2.4 config to   
 .config, run the following which will run through necessary   
 questions for the 2.6 kernel. Or, you might want to use the   
 config2.6-chirico...this has already been run through make   
 oldconfig on my system, and I've answered the necessary questions   
 for a general system.   
   
 make oldconfig   
   
 STEP 8:   
   
 This is very important. Make sure you're .config has the   
 following in it CONFIG\_EXT3\_FS=y You'll run into the following   
 error if you leave this =m instead of =y:   
   
 pivotroot: pivot\_root(/sysroot,/sysroot/initrd) failed   
   
 This is because Redhat 9.0 and 8.0 use the ext3 filesystem   
 for /boot ...   
   
   
 STEP 9:   
   
 Edit the Makefile and add changes to the Extraversion is needed.   
 Patches will update these values as well.   
   
 VERSION = 2   
 PATCHLEVEL = 6   
 SUBLEVEL = 3   
 EXTRAVERSION = -skim-ch6   
   
   
 STEP 10:   
   
 make bzImage   
   
   
 STEP 11:   
   
 make modules   
  
   
 STEP 12:   
   
 make modules\_install   
   
   
 STEP 13:   
   
 make install   
   
   
 If you come across errors here, what version of "depmod" is   
 being picked up in your path?   
   
 Also, if you get a module not found, say the following:   
 No module aic7xxx found for kernel 2.6.x   
 Then, in /lib/modules/2.6.x/kernel/drivers/scsi/aic7xxx/   
 cp aic7xxx.ko aic7xxx.o   
   
 insmod should look for aic7xxx.ko ;but , it looks for aic7xxx.o   
   
 If you still have trouble, make the following change in the   
 .config   
   
 CONFIG\_BLK\_DEV\_SD=y   
   
 and go back to STEP 10.   
   
 You also may want to ref   
 kernel-2.6.3-i686-smp-chirico-aic7xxx.config   
 in   
 http://prdownloads.sourceforge.net/souptonuts/configs-0.3.tar.gz?download   
   
 STEP 14:   
   
 mkdir /sys   
  
   
 STEP 15:   
   
 /etc/rc.sysinit needs to be modified. Look for the following   
 line:   
   
 action $"Mounting proc filesystem: "   
 mount -n -t proc /proc /proc   
   
   
 and after this line enter the following:   
   
 action $"Mounting sysfs filesystem: "   
 mount -t sysfs none /sys   
   
   
 Here's my /etc/rc.sysinit for reference:   
   
 http://prdownloads.sourceforge.net/souptonuts/rc.sysinit.txt?download   
   
   
   
 Be very careful at this step. Backup the /etc/rc.sysinit file.   
   
 Thomer [http://thomer.com/linux/migrate-to-2.6.html ] also added   
 changes to /etc/fstab. I only had to do STEP 16 below.   
   
   
   
 STEP 16:   
  
   
 Add the following to /etc/fstab for usb support.   
   
 /proc/bus/usb /proc/bus/usb usbdevfs defaults 0 0   
  
   
 STEP 17 (CHECKING EVERYTHING):   
   
 Check the following:   
   
 a. The new image file should be installed on boot and there   
 should be sym link to it. My latest kernel is 2.6.3-skim-ch6,   
 and I got the "-skim-ch6" from the values I put in the Makefile,   
 so I see the following:   
   
 /boot   
 vmlinuz -> vmlinuz-2.6.3-skim-ch6   
 System.map -> System.map-2.6.3-skim-ch6   
   
 /boot/grub/grub.conf Should have been automatically   
 updated from make.   
   
 In /boot/grub/grub.conf change "default=0" to boot   
 with the new kernel. Here's an example of my   
 grub.conf:   
   
   
 # grub.conf generated by anaconda   
 #   
 # Note that you do not have to rerun grub after making   
 # NOTICE: You have a /boot partition. This means that   
 # all kernel and initrd paths are relative to   
 # root (hd0,2)   
 # kernel /vmlinuz-version ro root=/dev/hda6   
 # initrd /initrd-version.img   
 #boot=/dev/hda   
 default=0   
 timeout=10   
 splashimage=(hd0,2)/grub/splash.xpm.gz   
 title Red Hat Linux (2.6.3-skim-ch6)   
 root (hd0,2)   
 kernel /vmlinuz-2.6.3-skim-ch6 ro root=LABEL=/   
 initrd /initrd-2.6.3-skim-ch6.img   
   
   
 b. The directory /sys exists   
   
 c. You added the mount command for sys in /etc/rc.sysinit   
   
 d. CONFIG\_EXT3\_FS=y was used in the .config   
   
 e. Run /sbin/lsmod or cat /proc/modules to make   
 sure a 2.4 kernel module wasn't forgotten. Also   
 look at "$cat /proc/iomem"   
   
   
   
 STEP 18 (DEVELOP YOUR OWN 2.6 MODULES):   
   
 You're done with the 2.6 build. So learn how to develop   
 2.6 kernel modules. First, checkout the following article   
   
 http://lwn.net/Articles/driver-porting/   
   
 Then, take a look at the following sample code, which shows how   
 to create /proc entries for communicating with the kernel and writing   
 out to any available tty device.   
   
 <http://prdownloads.sourceforge.net/souptonuts/procreadwrite.0.0.1a.tar.gz?download>   
   
   
   
   
 REFERENCES:   
   
 <http://www.codemonkey.org.uk/post-halloween-2.5.txt>   
 <http://kerneltrap.org/node/view/799>   
 <http://thomer.com/linux/migrate-to-2.6.html>   
 <http://www.kernel.org/>   
   
 <http://bugzilla.kernel.org/>   
 <http://groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=UTF-8&group=linux.kernel>   
 <http://linuxdevices.com/articles/AT3855888078.html>   
   
 <http://prdownloads.sourceforge.net/souptonuts/README_26.txt?download>   
 <http://prdownloads.sourceforge.net/souptonuts/rc.sysinit.txt?download>   
 <http://prdownloads.sourceforge.net/souptonuts/configs-0.3.tar.gz?download>   
 <https://sourceforge.net/forum/forum.php?forum_id=353715>   
   
 <http://www.redhat.com/software/rhel/kernel26/>   
 <http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO.html>   
 <http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO.html>   
   
 KERNEL DRIVER DEVELOPMENT IN 2.6:   
   
 Excellent (series of articles):   
 <http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO.html>   
   
 Here's my sample program:   
 <http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO.html>   
   
 Good but dated for 2.4 kernel:   
 <http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO.html>   
   
 <http://linuxdevices.com/articles/AT4389927951.html>   
 <http://linuxdevices.com/articles/AT5793467888.html>

======================================   
Setup Syslog-ng :   
======================================   
   
   
   
   
Hello. Here is roughly what you want to do. I don't have a box with elx on it, so I can't verify it 100%.   
  
Code:   
   
cd /var/tmp   
wget http://www.balabit.com/downloads/libol/0.3/libol-0.3.9.tar.gz   
tar xvfz libol-0.3.9.tar.gz   
cd libol-0.3.9   
./configure && make && make install   
   
wget http://www.balabit.com/downloads/syslog-ng/1.6/src/syslog-ng-1.6.7.tar.gz   
tar xvfz syslog-ng-1.6.7.tar.gz   
cd syslog-ng-1.6.7   
./configure && make && make install   
   
   
   
That gets syslog-ng on your system. One of the next things you have to do is get   
 syslog-ng.conf set up. By default, this file will be in /etc/syslog-ng/syslog-ng.conf.   
How you set it up depends entirely on what you want to do. Finally, you'll need to   
stop the old syslog from starting (should be a script in   
 /etc/rc.d) and get syslog-ng to start.   
   
   
======================================   
Setup CVS Repository :   
======================================   
   
   
Running a CVS Server   
Beyond Linux From Scratch - Version svn-20090409   
Running a CVS Server   
Running a CVS Server   
   
This section will describe how to set up, administer and secure a CVS server.   
CVS Server Dependencies   
Required   
   
CVS-1.11.23 and OpenSSH-5.1p1   
Setting up a CVS Server.   
   
A CVS server will be set up using OpenSSH as the remote access method. Other access   
methods, including :pserver: and :server: will not be used for write access to the   
CVS repository. The :pserver: method sends clear text passwords over the network   
and the :server: method is not supported in all CVS ports. Instructions for   
anonymous, read only CVS access using :pserver: can be found at the end of this   
section.   
   
Configuration of the CVS server consists of four steps:   
1. Create a Repository.   
   
Create a new CVS repository with the following commands:   
   
mkdir /srv/cvsroot && chmod 1777 /srv/cvsroot && export CVSROOT=/srv/cvsroot && cvs init   
   
2. Import Source Code Into the Repository.   
   
Import a source module into the repository with the following commands, issued from a  
user account on the same machine as the CVS repository:   
   
cd <sourcedir> && cvs import -m "<repository test>" <cvstest> <vendortag>   
<releasetag>   
   
3. Verify Local Repository Access.   
   
Test access to the CVS repository from the same user account with the following   
command:   
   
cvs co cvstest   
   
4. Verify Remote Repository Access.   
   
Test access to the CVS repository from a remote machine using a user account that has  
ssh access to the CVS server with the following commands:   
Note   
   
Replace <servername> with the IP address or host name of the CVS repository machine.   
You will be prompted for the user's shell account password before CVS checkout can   
continue.   
   
export CVS\_RSH=/usr/bin/ssh && cvs -d:ext:<servername>:/srv/cvsroot co cvstest   
   
Configuring CVS for Anonymous Read Only Access.   
   
CVS can be set up to allow anonymous read only access using the :pserver: method by   
logging on as root and executing the following commands:   
   
(grep anonymous /etc/passwd || useradd anonymous -s /bin/false -u 98) && echo   
 anonymous: > \   
/srv/cvsroot/CVSROOT/passwd && echo anonymous > /srv/cvsroot/CVSROOT/readers   
   
If you use inetd, the following command will add the CVS entry to /etc/inetd.conf:   
   
echo "2401 stream tcp nowait root /usr/bin/cvs cvs -f \   
 --allow-root=/srv/cvsroot pserver" >> /etc/inetd.conf   
   
Issue a killall -HUP inetd to reread the changed inetd.conf file.   
   
If you use xinetd, the following command will create the CVS file as   
 /etc/xinetd.d/cvspserver:   
   
cat >> /etc/xinetd.d/cvspserver << "EOF" # Begin /etc/xinetd.d/cvspserver   
service cvspserver   
{   
port = 2401   
socket\_type = stream   
protocol = tcp   
wait = no   
user = root   
passenv = PATH   
server = /usr/bin/cvs   
server\_args = -f --allow-root=/srv/cvsroot pserver }   
# End /etc/xinetd.d/cvspserver EOF   
   
Issue a /etc/rc.d/init.d/xinetd reload to reread the changed xinetd.conf file.   
   
Testing anonymous access to the new repository requires an account on another machine  
 that can reach the CVS server via network. No account on the CVS repository is   
 needed. To test anonymous access to the CVS repository, log in to another machine   
 as an unprivileged user and execute the following command:   
   
cvs -d:pserver:anonymous@<servername>:/srv/cvsroot co cvstest   
   
Note   
   
Replace <servername> with the IP address or hostname of the CVS server.   
Command Explanations   
   
mkdir /srv/cvsroot: Create the CVS repository directory.   
   
chmod 1777 /srv/cvsroot: Set sticky bit permissions for CVSROOT.   
   
export CVSROOT=/srv/cvsroot: Specify new CVSROOT for all cvs commands.   
   
cvs init: Initialize the new CVS repository.   
   
cvs import -m "repository test" cvstest vendortag releasetag: All source code modules  
 must be imported into the CVS repository before use, with the cvs import command.   
 The -m flags specifies an initial descriptive entry for the new module. The cvstest  
 parameter is the name used for the module in all subsequent cvs commands. The   
 vendortag and releasetag parameters are used to further identify each CVS module   
 and are mandatory whether used or not.   
   
(grep anonymous /etc/passwd || useradd anonymous -s /bin/false -u 98): Check for an   
 existing anonymous user and create one if not found.   
   
echo anonymous: > /srv/cvsroot/CVSROOT/passwd: Add the anonymous user to the CVS   
 passwd file, which is unused for anything else in this configuration.   
   
echo anonymous > /srv/cvsroot/CVSROOT/readers: Add the anonymous user to the CVS   
 readers file, a list of users who have read only access to the repository.   
Contents   
Installed Programs: None   
Installed Libraries: None   
Installed Directories: /srv/cvsroot   
   
Last updated on 2007-04-04 14:42:53 -0500   
  
 ======================================  
 Setup RSYNC :  
 ======================================

Beyond Linux From Scratch - Version svn-20090409   
rsync-3.0.2   
Introduction to rsync   
   
The rsync package contains the rsync utility. This is useful for synchronizing large   
 file archives over a network.  
   
Package Information   
   
 \* Download (HTTP):   
 http://anduin.linuxfromscratch.org/sources/BLFS/svn/r/rsync-3.0.2.tar.gz   
 \* Download MD5 sum: fd4c5d77d8cb7bb86ab209076fa214d9   
 \* Download size: 765 KB   
 \* Estimated disk space required: 35 MB (includes installing all documentation)   
 \* Estimated build time: 0.2 SBU   
   
rsync Dependencies   
Optional   
   
popt-1.14, libattr, and libacl   
   
User Notes: http://wiki.linuxfromscratch.org/blfs/wiki/rsync   
Installation of rsync   
   
For security reasons, running the rsync server as an unprivileged user and group is   
 encouraged. If you intend to run rsync as a daemon, create the rsyncd user and   
 group with the following commands issued by the root user:   
   
groupadd -g 48 rsyncd && useradd -c "rsyncd Daemon" -d /home/rsync -g rsyncd \ -s  
 /bin/false -u 48 rsyncd   
   
Install rsync by running the following commands:   
   
./configure --prefix=/usr && make   
   
If you have Doxygen-1.5.8 installed and wish to build HTML API documentation, issue   
 doxygen.   
   
If you have DocBook-utils-0.6.14 installed and wish to build the user documentation,   
 issue any or all of the following commands:   
   
pushd doc && docbook2pdf   
rsync.sgml && docbook2ps   
rsync.sgml && docbook2dvi   
rsync.sgml && docbook2txt   
rsync.sgml && docbook2html --nochunks rsync.sgml && popd   
   
To test the results, issue: make check.   
   
Now, as the root user:   
   
make install   
   
If you built the documentation, install it using the following commands as the root   
 user:   
   
install -v -m755 -d \   
/usr/share/doc/rsync-3.0.2/api && install -v -m644 dox/html/\* \   
/usr/share/doc/rsync-3.0.2/api && install -v \   
-m644 doc/rsync.\* /usr/share/doc/rsync-3.0.2   
   
Configuring rsync   
Config Files   
   
/etc/rsyncd.conf   
Configuration Information   
   
For client access to remote files, you may need to install the OpenSSH-5.1p1 package   
 to connect to the remote server.   
   
This is a simple download-only configuration to set up running rsync as a server. See  
 the rsyncd.conf(5) man-page for additional options (i.e., user authentication).   
   
cat > /etc/rsyncd.conf << "EOF"   
# This is a basic rsync configuration file   
# It exports a single module without user authentication.   
motd file = /home/rsync/welcome.msg   
use chroot = yes   
[localhost]   
path = /home/rsync   
comment = Default rsync module   
read only = yes   
list = yes   
uid = rsyncd   
gid = rsyncd EOF   
   
You can find additional configuration information and general documentation about   
 rsync at http://rsync.samba.org/documentation.html.  
   
Boot Script   
   
Note that you only want to start the rsync server if you want to provide an rsync   
 archive on your local machine. You don't need this script to run the rsync client.   
   
Install the /etc/rc.d/init.d/rsyncd init script included in the   
 blfs-bootscripts-20090302 package.   
   
make install-rsyncd   
   
Contents   
Installed Program: rsync   
Installed Libraries: None   
Installed Directories: Optionally, /usr/share/doc/rsync-3.0.2   
Short Descriptions   
   
rsync   
   
   
is a replacement for rcp (and scp) that has many more features. It uses the rsync   
 algorithm which provides a very fast method of syncing remote files. It does this   
 by sending just the differences in the files across the link, without requiring   
 that both sets of files are present at one end of the link beforehand.   
   
Last updated on 2008-08-11 09:59:48 -0500

### ====================================== 10 Linux and UNIX Interview Questions and Answers asked in Wipro TCS Capegemini

**UNIX and Linux Interview Questions and Answers**

Questions from various UNIX operating systems e.g. Solaris, Linux, IBM AIX or any other UNIX operating system is asked on different support and [programming interviews](http://javarevisited.blogspot.sg/2011/06/top-programming-interview-questions.html). I have always seen few interview questions from Linux and UNIX along with SQL in almost every Java programming interviews. You just can not afford not to prepare questions from UNIX and Linux until your Job absolutely doesn't require any work in UNIX operating system. I have collected many [UNIX command interview questions](http://javarevisited.blogspot.sg/2011/05/unix-command-interview-questions.html) and already shared them but I found that except system admin jobs, many programming job interviews only focus on general UNIX commands e.g. [chmod](http://javarevisited.blogspot.sg/2012/03/10-example-of-chmod-command-in-unix.html), [find](http://javarevisited.blogspot.sg/2011/03/10-find-command-in-unix-examples-basic.html) or [grep](http://javarevisited.blogspot.sg/2011/06/10-examples-of-grep-command-in-unix-and.html) and fundamentals like finding files and directories, managing file space, [networking commands](http://javarevisited.blogspot.sg/2010/10/basic-networking-commands-in-linuxunix.html), checking process status and managing file permissions. In this article we will see such kind of frequently asked interview questions and answers from UNIX and Linux operating System. Questions are very fundamental in nature and not limited to Linux only and equally applicable to other UNIX operating systems e.g. Solaris, IBM AIX etc. Many of these UNIX questions are asked during various interviews on companies like TCS, Infosys, Citibank , Wipro, Capegemini and Tech Mahindra. But as I said they are very fundamental and can be asked in any company.

## UNIX and Linux Interview questions answers

[Linux command interview questions and answers for programmers](http://javarevisited.blogspot.com/2012/06/10-xargs-command-example-in-linux-unix.html)Here is my list of frequently asked *UNIX and Linux interview questions and answers*. All these questions are based upon fundamental commands and concepts which is must for working in any UNIX operating system e.g. Solaris.

**1) How to find all the links in a folder in UNIX or Linux ?**

This is a *tricky UNIX question* as there is no specific command to find all symbolic links. Though you have [ln command for creating and updating soft links](http://javarevisited.blogspot.sg/2011/04/symbolic-link-or-symlink-in-unix-linux.html) but nothing which gives you all the links in a directory. You need to use ls command which list everything in directory and then you need to list all the links, as they starts with "l" as first characters, as shown in above article .

here is the actual UNIX command to find all links in a directory :

linux@nyj872:~ ls -lrt

total 2.0K

-rw-r--r-- 1 Linux Domain Users 0 Dec 6 2011 a

drwxr-xr-x+ 1 Linux Domain Users 0 Sep 19 12:30 java/

lrwxrwxrwx 1 Linux Domain Users 4 Sep 19 12:31 version\_1.0 -> java/

linux@nyj872:~ ls -lrt | grep '^l'

lrwxrwxrwx 1 Linux Domain Users 4 Sep 19 12:31 version\_1.0 -> java/

**2) How to find a process and kill that ?**

Another classic UNIX interview questions. Answer of this question is simple if you are familiar with ps, grep and kill command. by using "ps -ef" you can get list of all process and then use grep to find your process and get the PID of that process. Once you got PID you can use kill command to kill that process as shown in this [example of kill command in UNIX](http://javarevisited.blogspot.sg/2011/12/kill-command-unix-linux-example.html).

**3) How to run a program in background in UNIX or Linux ?**

an easy UNIX or Linux interview question, only when you know. You can use &amp; to run any process in background and than you can use jobs to find the job id for that process and can use fg and bg command to bring that process into foreground and background.

**4) How to sort output of a command in reverse order in Linux or UNIX ?**

One more Linux command interview question which checks knowledge of frequently used command. you can use sort command in UNIX to sort output of any command by using PIPE. By using -r option with sort command you can sort output of any command in reverse order. See these [sort command examples](http://javarevisited.blogspot.sg/2011/08/unix-sort-command-example-tutorial.html) for more details.

**5) How to create archive file in UNIX or Linux Operating System ?**

Another interview question based on knowledge of UNIX or Linux command. you can use [tar command](http://www.blogger.com/javarevisited.blogspot.sg/2011/11/tar-command-in-unix-linux-example.html) to great archives in UNIX or Linux. you can even combine tar and gzip to create a compressed archive in UNIX.

**6) What is meaning of a file has 644 permission ?**

To answer this UNIX or Linux interview question, you must know basics of files and directories in UNIX. 644 represents permission 110 for owner, permission 100 for group and 100 for others which means read + write for owner who create that file and read only permission for group and others. See this [tutorial on UNIX file permission](http://javarevisited.blogspot.sg/2011/11/file-permissions-in-unix-linux-example.html) for more details.

**7) How will you remove empty files or directories from /tmp ?**

See [how to delete empty directory and files in UNIX](http://javarevisited.blogspot.sg/2012/08/delete-empty-files-directories-unix.html) to answer this UNIX command interview questions.

**8) I have read permission on a directory but I am not able to enter it why ?**

One more tricky UNIX questions. In order to get into a directory you need execute permission. if your directory does not have execute permission than you can not go into that directory by using cd command. read [UNIX files and directory permissions](http://javarevisited.blogspot.sg/2011/11/file-permissions-in-unix-linux-example.html) for more information.

**9) How do you find all files which are modified 10 minutes before ?**

This is another the Linux interview questions from frequently used command e.g. find and grep. you can use -mtime option of find command to list all the files which are modified 10 or m minutes before. see these [find command examples](http://javarevisited.blogspot.sg/2011/03/10-find-command-in-unix-examples-basic.html) for more details.

**10) How to do you find size of directory in UNIX or Linux ?**

This is another tricky and bit tough Linux interview question as popular ls command doesn't show complete size of directories in UNIX. you need to use du command to get full size of directories including all sub directories in UNIX. See [How to find directory size in UNIX](http://javarevisited.blogspot.sg/2012/08/delete-empty-files-directories-unix.html) for exact command and detailed explanation.

======================================  
   
 System Management  
 User Management  
 Data Management  
 Veritas Clustering  
 Linux Clustering  
 More Scripting PHP Perl Expect SQL  
 ACLs  
 Oracle  
 MySQL  
 NMAP  
 More Mail  
 How to create a Solaris Package  
 How to work with Solaris Packages