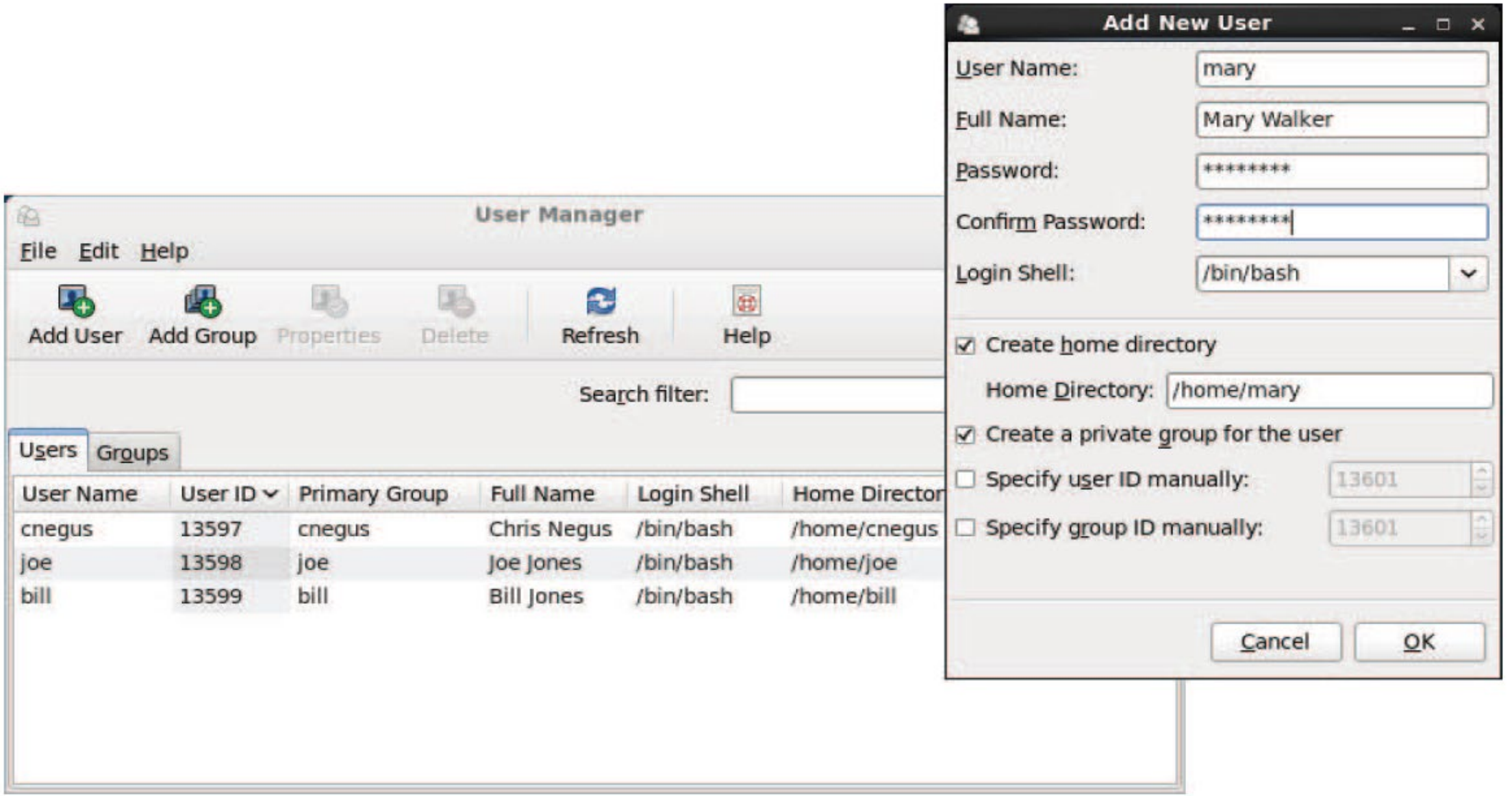
Linux for Devices

Module-4, Lecture-1

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

Managing User Accounts

- Creating User Accounts
  - ✓ Adding user with User Manager
  - ✓ Adding user with useradd
  - ✓ Modifying users with usermod
  - ✓ Deleting users with userdel


User Manager
File Edit Help
Add User
Add Group
Properties
Delete
Refresh
Help
Search filter:
Users
Groups
User Name
User ID
Primary Group
Full Name
Login Shell
Home Director
cnegus 13597 cnegus Chris Negus /bin/bash /home/cnegus
joe 13598 joe Joe Jones /bin/bash /home/joe
bill 13599 bill Bill Jones /bin/bash /home/bill
Add New User
User Name: mary
Full Name: Mary Walker
Password: ********
Confirm Password: ********
Login Shell: /bin/bash
Create home directory
Home Directory: /home/mary
Create a private group for the user
Specify user ID manually: 13601
Specify group ID manually: 13601
Cancel
OK

Managing User Accounts

- Understanding Group Accounts
  - ✓ Using group accounts
  - ✓ Creating group accounts
- Setting permissions with Access Control Lists
  - ✓ Setting ACLs with setfacl
  - ✓ Setting default ACLs
  - ✓ Enabling ACLs

- Adding directories for users to collaborate
  - ✓ Creating group collaboration directories (set GID bit)
  - ✓ Creating restricted deletion directories (sticky bit)

Name	Numeric value	Letter value
Set user ID bit	4	u+s
Set group ID bit	2	g+s
Sticky bit	1	o+t

Linux for Devices

Module-4, Lecture-2

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

Managing Disks

- Understanding Disk Storage
  - ✓Partitioning Hard Disks
  - ✓Viewing disk partitions
  - ✓Creating a single-partition disk
  - ✓Creating a multiple-partition disk

Managing Disks

- Using Logical Volume Management Partitions
  - ✓Checking an existing LVM
  - ✓Creating LVM logical volumes
  - ✓Growing LVM logical volumes

Managing Disks

- Mounting Filesystems
  - ✓Supported filesystems
  - ✓Enabling swap areas
  - ✓Disabling swap area
  - ✓Using the fstab file to define mountable file systems
  - ✓Using the mount command to mount file systems
  - ✓Mounting a disk image in loopback
  - ✓Using the umount command
  - ✓Using the mkfs Command to Create a Filesystem

```
# fdisk -c -u -l /dev/sdc
Disk /dev/sdc: 8021 MB, 8021606400 bytes
16 heads, 48 sectors/track, 20400 cylinders, total 15667200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc3072e18
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048     4196351     2097152   83  Linux
/dev/sdc2         4196352     5220351      512000   82  Linux swap / Solaris
/dev/sdc3         5220352     7317503     1048576    c  W95 FAT32 (LBA)
/dev/sdc4         7317504    15667199     4174848    5  Extended
/dev/sdc5         7319552     7729151      204800   83  Linux
/dev/sdc6         7731200    15667199     3968000   8e  Linux LVM
```

- Your first primary hard disk usually appears as /dev/sda.
- With RHEL and Fedora installations, there is usually at least one LVM partition, out of which other partitions can be assigned.
- So, the output of fdisk might be as simple as the following:

```
# fdisk -cul /dev/sda
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ebb20
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      411647      204800   83  Linux
/dev/sda2          411648   976773119   488180736   8e  Linux LVM
```

Creating a single-partition disk

The general procedure:

1. Install the new hard drive or insert the new USB flash drive.
2. Partition the new disk.
3. Create the filesystems on the new disk.
4. Mount the filesystems

- The following process takes you through partitioning a USB flash drive to be used for Linux that has only one partition

- For a USB flash drive, just plug it into an available USB port.
- Determine the device name for the hard disk: # tail -f /var/log/messages

```
# tail -f /var/log/messages
scsi 6:0:0:0: Direct-Access S31B1102 USB DISK 1100 PQ: 0 ANSI: 0 CCS
sd 6:0:0:0: Attached scsi generic sg2 type 0
sd 6:0:0:0: [sdc] 8342528 512-byte logical blocks: (4.27 GB/3.97 GiB)
sd 6:0:0:0: [sdc] Write Protect is off
sd 6:0:0:0: [sdc] Mode Sense: 43 00 00 00
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Assuming drive cache: write through
 sdc: sdc1
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Attached SCSI removable disk
```

- From the output, you can see that the USB flash drive was found and assigned to /dev/sdc etc.
- If the USB flash drive mounts automatically, unmount it: umount /dev/sdc1
- Use the fdisk command to create partitions on the new disk.

```
# fdisk -c -u /dev/sdc
Command (m for help):
```

- If you start with a new USB flash drive, it may have one partition that is entirely devoted to a Windows-compatible filesystem (such as VFAT).
- Use p to view all partitions and d to delete the partition

```
Command (m for help): p
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048    15667199     7832576    c  W95 FAT32 (LBA)
Command (m for help): d
Selected partition 1
```

- To create a new partition, type the letter n. You are prompted for the type of partition
- Choose an extended (e) or primary partition (p). Type the letter p to choose primary.

- Type in the partition number. If you are creating the first partition (or for only one partition), type the number 1. You are prompted for the first sector to start the partition.
- Select the first available sector number (you can just press enter to choose it). You are prompted for the last sector.
- Enter the size of the partition. Because you are just creating one partition to consume the whole disk, choose the last available sector. To do that you can just press Enter to accept the default

- Double-check that the drive is partitioned the way you want by pressing p.

```
Command (m for help): p
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    15667199     7832576   83  Linux
```

- To make changes to the partition table permanent, type w. This will write the changes, try to sync those changes with the Linux kernel, and quit fdisk.
- If fdisk cannot sync the partition table on the disk with the kernel, the most likely reason is that a partition from the disk is still mounted.

- Unmount the partition, and then try running the following command to sync the disk partition table with the kernel: # partprobe /dev/sdc
- If partprobe does not work, rebooting the computer will make sure the disk and kernel are in sync.
- Although the partitioning is done, the new partition is not yet ready to use. For that, you must create a filesystem on the new partition:

mkfs -t ext4 /dev/sdc1

- To be able to use the new filesystem, you need to create a mount point and mount it to the partition.

```
# mkdir /mnt/test
# mount /dev/sdc1 /mnt/test
# df -h /mnt/test
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdc1             7.4G   17M  7.0G   1% /mnt/test
# mount | grep sdc1
/dev/sdc1 on /mnt/test type ext4 (rw)
```

- When you are done using the drive, you can unmount it with the umount command, after which you can safely remove the drive.

umount /dev/sdc1

- Set up a USB flash drive to mount automatically every time the system boots.

  Edit /etc/fstab and add a line describing what and where to mount. A line you might add: /dev/sdc1 /mnt/test ext4 defaults 0 1

Managing User Accounts

- Understanding Group Accounts
  - ✓ Using group accounts
  - ✓ Creating group accounts
- Setting permissions with Access Control Lists
  - ✓ Setting ACLs with setfacl
  - ✓ Setting default ACLs
  - ✓ Enabling ACLs

- Adding directories for users to collaborate
  - ✓ Creating group collaboration directories (set GID bit)
  - ✓ Creating restricted deletion directories (sticky bit)

Name	Numeric value	Letter value
Set user ID bit	4	u+s
Set group ID bit	2	g+s
Sticky bit	1	o+t

Linux for Devices

Module-4, Lecture-3

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

Managing Filesystem

- Understanding Disk Storage
  - ✓ Partitioning Hard Disks
  - ✓ Viewing disk partitions
  - ✓ Creating a single-partition disk
  - ✓ Creating a multiple-partition disk

Managing Disks

- Using Logical Volume Management Partitions
  - ✓Checking an existing LVM
  - ✓Creating LVM logical volumes
  - ✓Growing LVM logical volumes

Managing Disks

- Mounting Filesystems
  - ✓Supported filesystems
  - ✓Enabling swap areas
  - ✓Disabling swap area
  - ✓Using the fstab file to define mountable file systems
  - ✓Using the mount command to mount file systems
  - ✓Mounting a disk image in loopback
  - ✓Using the umount command
  - ✓Using the mkfs Command to Create a Filesystem

```
# fdisk -c -u -l /dev/sdc
Disk /dev/sdc: 8021 MB, 8021606400 bytes
16 heads, 48 sectors/track, 20400 cylinders, total 15667200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc3072e18
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048     4196351     2097152   83  Linux
/dev/sdc2         4196352     5220351      512000   82  Linux swap / Solaris
/dev/sdc3         5220352     7317503     1048576    c  W95 FAT32 (LBA)
/dev/sdc4         7317504    15667199     4174848    5  Extended
/dev/sdc5         7319552     7729151      204800   83  Linux
/dev/sdc6         7731200    15667199     3968000   8e  Linux LVM
```

- Your first primary hard disk usually appears as /dev/sda.
- With RHEL and Fedora installations, there is usually at least one LVM partition, out of which other partitions can be assigned.
- So, the output of fdisk might be as simple as the following:

```
# fdisk -cul /dev/sda
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ebb20
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      411647      204800   83  Linux
/dev/sda2          411648   976773119   488180736   8e  Linux LVM
```

Creating a single-partition disk

The general procedure:

1. Install the new hard drive or insert the new USB flash drive.
2. Partition the new disk.
3. Create the filesystems on the new disk.
4. Mount the filesystems

- The following process takes you through partitioning a USB flash drive to be used for Linux that has only one partition

- For a USB flash drive, just plug it into an available USB port.
- Determine the device name for the hard disk: # tail -f /var/log/messages

```
# tail -f /var/log/messages
scsi 6:0:0:0: Direct-Access S31B1102 USB DISK 1100 PQ: 0 ANSI: 0 CCS
sd 6:0:0:0: Attached scsi generic sg2 type 0
sd 6:0:0:0: [sdc] 8342528 512-byte logical blocks: (4.27 GB/3.97 GiB)
sd 6:0:0:0: [sdc] Write Protect is off
sd 6:0:0:0: [sdc] Mode Sense: 43 00 00 00
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Assuming drive cache: write through
 sdc: sdc1
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Attached SCSI removable disk
```

- From the output, you can see that the USB flash drive was found and assigned to /dev/sdc etc.
- If the USB flash drive mounts automatically, unmount it: umount /dev/sdc1
- Use the fdisk command to create partitions on the new disk.

```
# fdisk -c -u /dev/sdc
Command (m for help):
```

- If you start with a new USB flash drive, it may have one partition that is entirely devoted to a Windows-compatible filesystem (such as VFAT).
- Use p to view all partitions and d to delete the partition

```
Command (m for help): p
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048    15667199     7832576    c  W95 FAT32 (LBA)
Command (m for help): d
Selected partition 1
```

- To create a new partition, type the letter n. You are prompted for the type of partition
- Choose an extended (e) or primary partition (p). Type the letter p to choose primary.

- Type in the partition number. If you are creating the first partition (or for only one partition), type the number 1. You are prompted for the first sector to start the partition.
- Select the first available sector number (you can just press enter to choose it). You are prompted for the last sector.
- Enter the size of the partition. Because you are just creating one partition to consume the whole disk, choose the last available sector. To do that you can just press Enter to accept the default

- Double-check that the drive is partitioned the way you want by pressing p.

```
Command (m for help): p
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    15667199     7832576   83  Linux
```

- To make changes to the partition table permanent, type w. This will write the changes, try to sync those changes with the Linux kernel, and quit fdisk.
- If fdisk cannot sync the partition table on the disk with the kernel, the most likely reason is that a partition from the disk is still mounted.

- Unmount the partition, and then try running the following command to sync the disk partition table with the kernel: # partprobe /dev/sdc
- If partprobe does not work, rebooting the computer will make sure the disk and kernel are in sync.
- Although the partitioning is done, the new partition is not yet ready to use. For that, you must create a filesystem on the new partition:

mkfs -t ext4 /dev/sdc1

- To be able to use the new filesystem, you need to create a mount point and mount it to the partition.

```
# mkdir /mnt/test
# mount /dev/sdc1 /mnt/test
# df -h /mnt/test
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdc1             7.4G   17M  7.0G   1% /mnt/test
# mount | grep sdc1
/dev/sdc1 on /mnt/test type ext4 (rw)
```

- When you are done using the drive, you can unmount it with the umount command, after which you can safely remove the drive.

umount /dev/sdc1

- Set up a USB flash drive to mount automatically every time the system boots.

  Edit /etc/fstab and add a line describing what and where to mount. A line you might add: /dev/sdc1 /mnt/test ext4 defaults 0 1

Linux for Devices

Module-4, Lecture-4

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

Challenges as a server administrator:

- ✓ Remote access
- ✓ Diligent security
- ✓ Continuous monitoring

- Starting with Server Administration
  - ✓Step 1: Install the server
  - ✓Step 2: Configure the server
  - ✓Step 3: Start the server
  - ✓Step 4: Secure the server
  - ✓Step 5: Monitor the server

- Managing Remote Access with the Secure Shell Service
  - ✓Starting the openssh-server service
  - ✓Using SSH client tools
  - ✓Using key-based (passwordless) authentication

- Configuring System Logging
  - ✓Enabling system logging with rsyslog
  - ✓Watching logs with logwatch
- Checking System Resources with sar
- Checking System Space
  - ✓Displaying system space with df
  - ✓Checking disk usage with du

TABLE 13.1: Commands to Determine sshd Status

Distribution	Command to Determine sshd Status
RHEL	chkconfig --list sshd
Fedora	systemctl status sshd.service
Ubuntu	status ssh

TABLE 13.2: Commands to Start sshd

Distribution	Command to Start sshd
RHEL	service sshd start
Fedora	systemctl start sshd.service
Ubuntu	service ssh start

TABLE 13.3: Commands to Start sshd at Boot

Distribution	Command to Start sshd at Boot
RHEL	chkconfig sshd on
Fedora	systemctl enable sshd.service
Ubuntu	update-rc.d ssh defaults

- The following is an example of remotely logging in to johndoe's account on 10.140.67.23:

```
$ ssh johndoe@10.140.67.23
The authenticity of host '10.140.67.23 (10.140.67.23)' can't be established.
RSA key fingerprint is a4:28:03:85:89:6d:08:fa:99:15:ed:fb:b0:67:55:89.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.140.67.23' (RSA) to the list of known hosts.
johndoe@10.140.67.23's password: *********
```

Linux for Devices

Module-4, Lecture-5

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

- Type of Networks
  - ✓Wired network
  - ✓Wireless network
- The general approach to configuring networking in three types of Linux systems
  - ✓Desktop/laptop networking
  - ✓Server networking
  - ✓Enterprise networking

Group	Title	Description
ISOC	Internet Society	Professional membership organization of Internet experts that oversees boards and task forces dealing with network policy issues **isoc.org**
IESG	The Internet Engineering Steering Group	Responsible for technical management of IETF activities and the Internet standards process **ietf.org/iesg.html**
IANA	Internet Assigned Numbers Authority	Responsible for Internet Protocol (IP) addresses **iana.org**
IAB	Internet Architecture Board	Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF **iab.org**
IETF	Internet Engineering Task Force	Protocol engineering and development arm of the Internet **ietf.org**

TABLE 34-1 TCP/IP Protocol Development Groups

TCP/IP Protocol Suite

- The three basic protocols are the Transmission Control Protocol (TCP), which handles receiving and sending out communications;
- The Internet Protocol (IP), which handles the actual transmissions; and
- The User Datagram Protocol (UDP), which also handles receiving and sending packets.
- The IP protocol, which is the base protocol that all others use, handles the actual transmissions, the packets of data with sender and receiver information in each.

- The TCP protocol is designed to work with cohesive messages or data.
- This protocol checks received packets and sorts them into their designated order, forming the original message.
- For data sent out, the TCP protocol breaks the data into separate packets, designating their order.
- The UDP protocol, meant to work on a much more raw level, also breaks down data into packets but does not check their order.

- The TCP/IP protocol is designed to provide stable and reliable connections that ensure that all data is received and reorganized into its original order
- UDP, on the other hand, is designed to simply send as much data as possible, with no guarantee that packets will all be received or placed in the proper order.
- UDP is often used for transmitting very large amounts of data of the type that can survive the loss of a few packets—for example, temporary images, video, and banners displayed on the Internet.

- Other protocols provide various network and user services. The Domain Name Service (DNS) provides address resolution.
- The File Transfer Protocol (FTP) provides file transmission, and the Network File System (NFS) provides access to remote file systems.
- Table 34-2 lists the different protocols in the TCP/IP protocol suite.
- These protocols make use of either the TCP or UDP protocol to send and receive packets, which, in turn, uses the IP protocol for actually transmitting the packets.

Transport	Description
TCP	Transmission Control Protocol; places systems in direct communication
UDP	User Datagram Protocol
IP	Internet Protocol; transmits data
ICMP	Internet Control Message Protocol; status messages for IP
Routing	**Description**
RIP	Routing Information Protocol; determines routing
OSPF	Open Shortest Path First; determines routing
Network Address	**Description**
ARP	Address Resolution Protocol; determines unique IP address of systems
DNS	Domain Name Service; translates hostnames into IP addresses
RARP	Reverse Address Resolution Protocol; determines addresses of systems

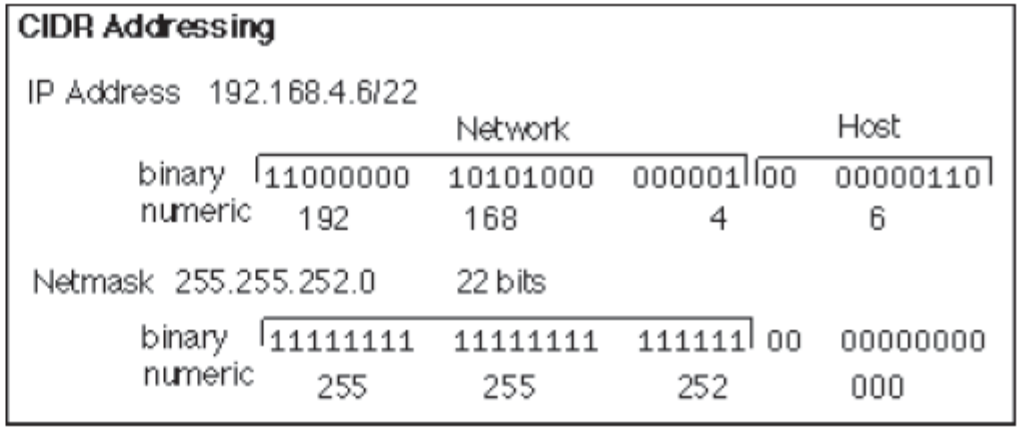
User Service	Description
FTP	File Transfer Protocol; transmits files from one system to another using TCP
TFTP	Trivial File Transfer Protocol; transfers files using UDP
Telnet	Remote login to another system on the network
SMTP	Simple Mail Transfer Protocol; transfers email between systems
RPC	Remote Procedure Call; allows programs on remote systems to communicate
Gateway	**Description**
EGP	Exterior Gateway Protocol; provides routing for external networks
GGP	Gateway-to-Gateway Protocol; provides routing between Internet gateways
IGP	Interior Gateway Protocol; provides routing for internal networks

Network Service	Description
NFS	Network File System; allows mounting of file systems on remote machines
NIS	Network Information Service; maintains user accounts across a network
BOOTP	Boot Protocol; starts system using boot information on server for network
SNMP	Simple Network Management Protocol; provides status messages on TCP/IP configuration
DHCP	Dynamic Host Configuration Protocol; automatically provides network configuration information to host systems

TABLE 34-2 TCP/IP Protocol Suite

- Configuring Networks on GNOME and KDE
- Zero Configuration Networking (zeroconf): Avahi and Link Local Addressing
- IPv4 and IPv6
- TCP/IP Network Addresses
- IPv4 Network Addresses
- Class-Based IP Addressing
- Netmask
- Classless Interdomain Routing (CIDR)

FIGURE 34-2
CIDR addressing



- IPv4 CIDR Addressing
- IPv6 CIDR Addressing

Short Form	Full Form	Maximum Number of Hosts
/8	/255.0.0.0	16,777,215 (A class)
/16	/255.255.0.0	65,534 (B class)
/17	/255.255.128.0	32,767
/18	/255.255.192.0	16,383
/19	/255.255.224.0	8191
/20	/255.255.240.0	4095
/21	/255.255.248.0	2047
/22	/255.255.252.0	1023
/23	/255.255.254.0	511
/24	/255.255.255.0	255 (C class)
/25	/255.255.255.128	127
/26	/255.255.255.192	63
/27	/255.255.255.224	31
/28	/255.255.255.240	15
/29	/255.255.255.248	7
/30	/255.255.255.252	3

TABLE 34-3 CIDR IPv4 Network Masks

Subnetwork	CIDR Address	Binary Mask
First subnet network address	.0/25	00000000
Second subnet network address	.128/25	10000000
First subnet broadcast address	.127/25	01111111
Second subnet broadcast address	.255/25	11111111
First address in first subnet	.1/25	00000001
First address in second subnet	.129/25	10000001
Last address in first subnet	.126/25	01111110
Last address in second subnet	.254/25	11111110

Linux for Devices

Module-4, Lecture-6

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

- Understanding the Linux init Daemon
  - Understanding the classic init daemons
  - Understanding the Upstart init daemon
  - Understanding systemd init
- Auditing Services
  - Auditing the classic SysVinit daemon
  - Auditing the Upstart init daemon
  - Auditing the systemd init

- Stopping and Starting Services
  - Stopping and starting the classic SysVinit daemon
  - Stopping and starting the Upstart init daemon
  - Stopping and starting the systemd daemon
- Configuring Persistent Services
  - Configuring the classic SysVinit daemon persistent services
  - Configuring Upstart init daemon persistent services
  - Configuring systemd init persistent services

- Configuring a Default runlevel or target unit
  - Configuring the classic SysVinit daemon default runlevel
  - Configuring the Upstart init daemon default runlevel
  - Configuring the systemd init default target unit
- Adding New or Customized Services
  - Adding new services to classic SysVinit daemon
  - Adding new services to the Upstart init daemon
  - Adding new services to systemd init

Stopping and starting the classic SysVinit daemon

- The primary command for stopping and starting SysVinit services is the service command.
- With the service command, the name of the service you are wishing to control comes second in the command line.
- The last option is what you want to do to the service, stop, start, restart, and so on.
- The following example shows how to stop the cups daemon.

- Notice that an OK is given, which lets you know that cupsd has been successfully stopped.

```
# service cups status
cupsd (pid 5857) is running...
#
# service cups stop
Stopping cups:                [  OK  ]
#
# service cups status
cupsd is stopped
```

- To start a service, you simply add a start option instead of a stop option on the end of the service command as follows:

```
# service cups start
Starting cups:              [  OK  ]
#
# service cups status
cupsd (pid 6860) is running...
```

- To restart a SysVinit service, the restart option is used. This option will stop the service and then immediately start it again.

```
# service cups restart
Stopping cups:                [  OK  ]
Starting cups:                [  OK  ]
#
# service cups status
cupsd (pid 7955) is running...
```

- When a service is already stopped, a restart will generate a FAILED status on the attempt to stop it. However, as shown in the example that follows, the service will be successfully started when a restart is attempted.

```
# service cups stop
Stopping cups:                [  OK  ]
#
# service cups restart
Stopping cups:                [FAILED]
Starting cups:                [  OK  ]
#
# service cups status
cupsd (pid 8236) is running...
```

- Reloading a service is different from restarting a service. When you reload a service, the service itself is not stopped. Only the service's configuration files are loaded again. The following example shows how to reload the cups daemon.

```
# service cups status
cupsd (pid 8236) is running...
#
# service cups reload
Reloading cups:                    [  OK  ]
#
# service cups status
cupsd (pid 8236) is running...
```

- If a SysVinit service is stopped when you attempt to reload it, you will get a FAILED status. This is shown in the following example:

```
# service cups status
cupsd is stopped
#
# service cups reload
Reloading cups:                    [FAILED]
```

Stopping and starting the Upstart init daemon

- The primary command for stopping and starting Upstart init services is the initctl command. The options are very similar to SysVinit's service command:
- Stopping a service with Upstart init — In the following example, the status of the cups daemon is checked and then stopped using the initctl stop cups.service command.

```
# initctl status cups
cups start/running, process 2390
#
# initctl stop cups
cups stop/waiting
#
# initctl status cups
cups stop/waiting
```

- Starting a service with Upstart init — In the following example, the cups daemon is stopped using the initctl stop cups.service command.

```
# initctl start cups
cups start/running, process 2408
#
# initctl status cups
cups start/running, process 2408
```

- Restarting a service with Upstart init — Restarting a service with Upstart init will stop and then start the service. However, the configuration file will not be reloaded.

```
# initctl restart cups
cups start/running, process 2490
#
# initctl status cups
cups start/running, process 2490
#
```

- Reloading a service with Upstart init — Reloading will not stop and start the service. It only loads the configuration file again. This is the option to use when you have made changes to the configuration file.

```
# initctl reload cups
#
# initctl status cups
cups start/running, process 2490
```

- Notice that the process ID (PID) is still 2490, which is the same as it was in the example for restarting the cups daemon because the process was not stopped and started in the reload process.

Stopping and starting the systemd daemon

- For the systemd daemon, the systemctl command will work for stopping, starting, reloading, and restarting. The options to the systemctrl command should look familiar.
- Stopping a service with system - In the example that follows, the status of the cups daemon is checked and then stopped using the systemctl stop cups.service command.
- Notice that when the status is taken, after stopping the cups daemon, the service is inactive (dead) but still considered enabled. This means that the cups daemon will still be started upon server boot.

```
# systemctl status cups.service
cups.service - CUPS Printing Service
    Loaded: loaded (/lib/systemd/system/cups.service; enabled)
    Active: active (running) since Mon, 30 Apr 2015 12:36:3...
  Main PID: 1315 (cupsd)
    CGroup: name=systemd:/system/cups.service
            └ 1315 /usr/sbin/cupsd -f
#
# systemctl stop cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
    Loaded: loaded (/lib/systemd/system/cups.service; enabled)
    Active: inactive (dead) since Tue, 01 May 2015 04:43:4...
    Process: 1315 ExecStart=/usr/sbin/cupsd -f
 (code=exited, status=0/SUCCESS)
    CGroup: name=systemd:/system/cups.service
```

- Starting a service with system - Starting the cups daemon is just as easy as stopping it. The example that follows demonstrates this ease.

```
# systemctl start cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
    Loaded: loaded (/lib/systemd/system/cups.service; enabled)
    Active: active (running) since Tue, 01 May 2015 04:43:5...
  Main PID: 17003 (cupsd)
    CGroup: name=systemd:/system/cups.service
            └ 17003 /usr/sbin/cupsd -f
```

- After the cups daemon is started, using systemctl with the status option shows the service is active (running). Also, its process ID (PID) number, 17003, is shown

- Restarting a service with system - Restarting a service means that a service is stopped and then started again. If the service was not currently running, restarting it will simply start the service.

```
# systemctl restart cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
   Loaded: loaded (/lib/systemd/system/cups.service; enabled)
   Active: active (running) since Tue, 01 May 2015 04:45:2...
  Main PID: 17015 (cupsd)
   CGroup: name=systemd:/system/cups.service
           └ 17015 /usr/sbin/cupsd -f
```

- You can also perform a conditional restart of a service using systemctl. A conditional restart only restarts a service if it is currently running. Any service in an inactive state will not be started.
- Notice in the example that the cups daemon was in an inactive state. When the conditional restart was issued, no error messages were generated! The cups daemon was not started because conditional restarts will affect active services.
- Thus, it is always a good practice to check the status of a service, after stopping, starting, conditionally restarting, and so on.

```
# systemctl status cups.service
cups.service - CUPS Printing Service
  Loaded: loaded (/lib/systemd/system/cups.service; enabled)
  Active: inactive (dead) since Tue, 01 May 2015 06:03:32...
 Process: 17108 ExecStart=/usr/sbin/cupsd -f
 (code=exited, status=0/SUCCESS)
  CGroup: name=systemd:/system/cups.service
#
# systemctl condrestart cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
  Loaded: loaded (/lib/systemd/system/cups.service; enabled)
  Active: inactive (dead) since Tue, 01 May 2015 06:03:32...
 Process: 17108 ExecStart=/usr/sbin/cupsd -f
 (code=exited, status=0/SUCCESS)
  CGroup: name=systemd:/system/cups.service
```

- Reloading a service with system - Reloading a service is different from restarting a service. When you reload a service, the service itself is not stopped. Only the service's configuration files are loaded again
- Doing a reload of a service, instead of a restart, will prevent any pending service operations from being aborted.
- A reload is a better method for a busy Linux server.
- Now that you know how to stop and start services for troubleshooting and emergency purposes, you can learn how to enable and disable services

```
# systemctl reload cups.service
Failed to issue method call: Job type reload is
 not applicable for unit cups.service.
#
# systemctl reload sshd.service
#
# systemctl status sshd.service
sshd.service - OpenSSH server daemon
    Loaded: loaded (/lib/systemd/system/sshd.service; enabled)
    Active: active (running) since Mon, 30 Apr 2015 12:35:2...
    Process: 17009 ExecReload=/bin/kill -HUP $MAINPID
 (code=exited, status=0/SUCCESS)
  Main PID: 786 (sshd)
    CGroup: name=systemd:/system/sshd.service
            └ 786 /usr/sbin/sshd -D
```

Configuring an NFS File Server

Department of Computer Science and Engineering

Linux for Devices

Module-4, Lecture-7

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

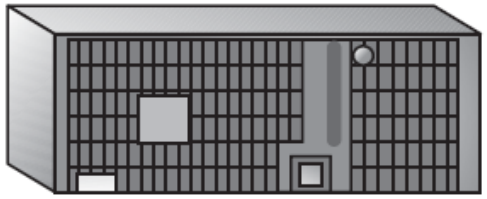
- Configuring an NFS File Server
  - ✓Installing an NFS Server
  - ✓Starting the NFS service
  - ✓Sharing NFS Filesystems
  - ✓Configuring the /etc/exports file
  - ✓Exporting the shared filesystems
  - ✓Securing Your NFS Server
  - ✓Opening your firewall for NFS
  - ✓Allowing NFS access in TCP wrappers
  - ✓Configuring SELinux for your NFS server
  - ✓Using NFS Filesystems
  - ✓Mounting an NFS filesystem at boot time
  - ✓Using autofs to mount NFS filesystems on demand
  - ✓Unmounting NFS filesystems

An NFS file server provides an easy way to share large amounts of data among the users and computers in an organization. An administrator of a Linux system that is configured to share its filesystems using NFS has to perform the following tasks to set up NFS:

1. Set up the network
2. Start the NFS service.
3. Choose what to share from the server
4. Set up security on the server
5. Mount the filesystem on the client

/etc/exports file:
/apps/bin pins(rw) maple(rw) spruce(rw)

 # mount -t nfs oak:/apps/bin /oak/apps



oak

/
apps/
bin/
file1 file2 file3

pine

/
oak/
apps/

Installing an NFS Server

- # yum install nfs-utils
- To find out more about the nfs-utils package, you can run the following commands to see information about the package, configuration files, and commands, respectively:
  - # rpm -qi nfs-utils
  - # rpm -qc nfs-utils
  - # rpm -ql nfs-utils | grep bin

Starting the NFS service

- The basic NFS service in Fedora is called nfs-server.

```
# systemctl start nfs-server.service
# systemctl enable nfs-server.service
# systemctl status nfs-server.service
systemctl status nfs-server.service
nfs-server.service - NFS Server
     Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled)
     Active: active (running) since Sat, 2 Jun 2012 08:40:25 -0400;
             1h 28min ago
   Main PID: 7767 (rpc.mountd)
     CGroup: name=systemd:/system/nfs-server.service
             └ 7767 /usr/sbin/rpc.mountd
```

- In Red Hat Enterprise Linux 6, you need the service and chkconfig commands to check, start, and enable the NFS service (nfs).
- The following commands show the nfs service not running currently and disabled:

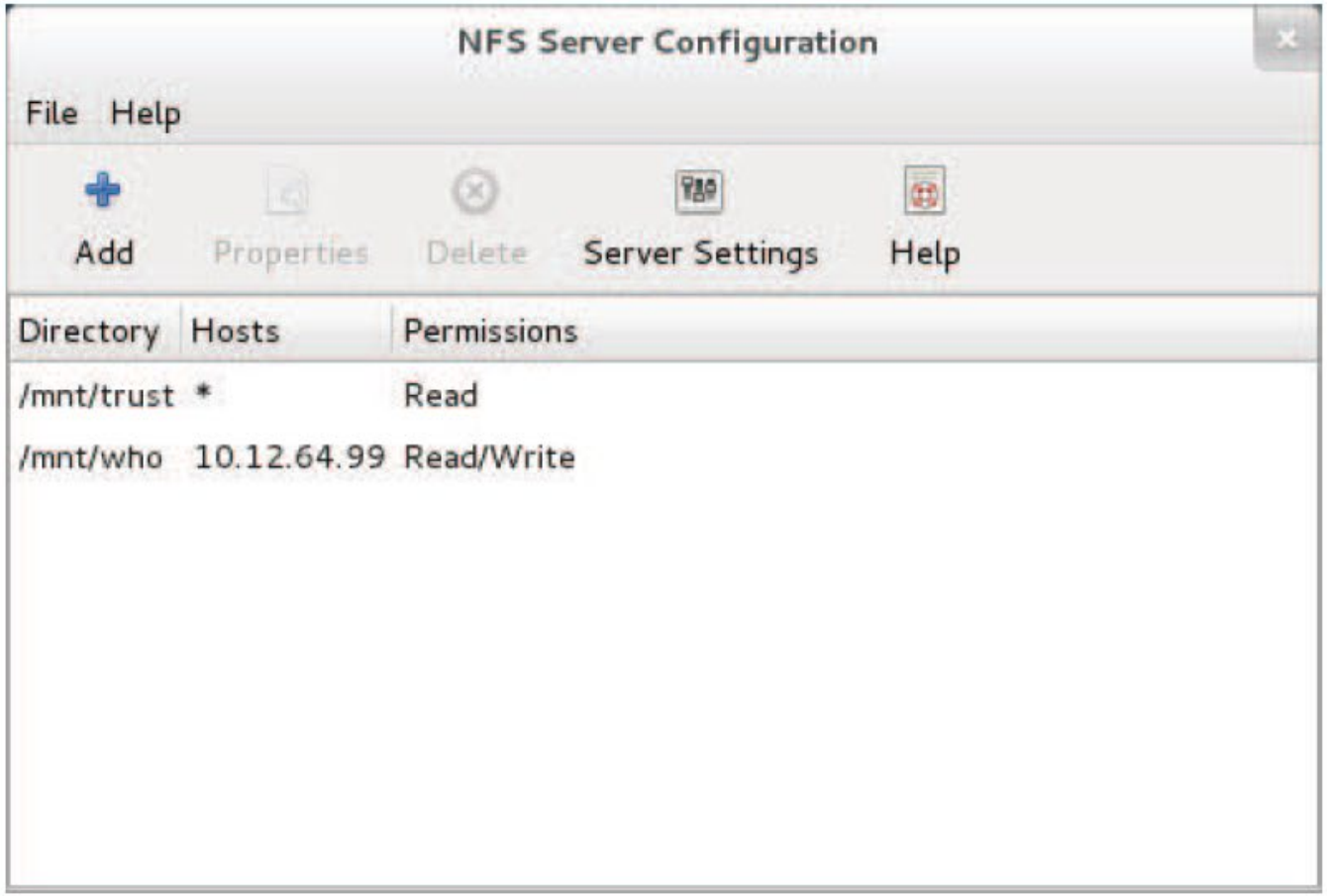
```
# service nfs status
rpc.svcgssd is stopped
rpc.mountd is stopped
nfsd is stopped
# chkconfig --list nfs
nfs   0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

- As mentioned earlier, the rpcbind service must be running for NFS to work. So, you could use the following commands to start and permanently enable both the rpcbind and nfs services

```
# service rcpbind start
Starting rpcbind:                          [  OK  ]
# service nfs start
Starting NFS services:                     [  OK  ]
Starting NFS quotas:                       [  OK  ]
Starting NFS daemon:                       [  OK  ]
Starting NFS mountd:                       [  OK  ]
# chkconfig nfs on
```

Sharing NFS Filesystems

The NFS Server Configuration window (`system-config-nfs`) provides a graphical way of configuring NFS services.



- To share an NFS filesystem from your Linux system, you need to export it from the server system.
- Exporting is done in Linux by adding entries into the /etc/exports file.
- Each entry identifies a directory in your local filesystem that you want to share with other computers.
- The entry also identifies the other computers that can share the resource (or opens it to all computers) and includes other options that reflect permissions associated with the directory.

- Remember that when you share a directory, you are sharing all files and subdirectories below that directory as well (by default). So, you need to be sure that you want to share everything in that directory structure.
- There are still ways to restrict access within that directory structure

Configuring the /etc/exports file

- To make a directory from your Linux system available to other systems, you need to export that directory. Exporting is done on a permanent basis by adding information about an exported directory to the /etc/exports file. The format of the /etc/exports file is:

```
Directory   Host(Options...)   Host(Options...)   # Comments
```

- where Directory is the name of the directory that you want to share and Host indicates the client computer to which the sharing of this directory is restricted.
- Options can include a variety of options to defi ne the security measures attached to the shared directory for the host.
- Comments are any optional comments you want to add (following the # sign).

- As root user, you can use any text editor to configure /etc/exports to modify shared directory entries or add new ones. Here's an example of an /etc/exports file:

```
/cal     *.linuxtoys.net(rw)                  # Company events
/pub     *(ro,insecure,all_squash)            # Public dir
/home    maple(rw,root_squash) spruce(rw,root_squash)
```

Exporting the shared filesystems

Here's an example of the `exportfs` command:

```
# /usr/sbin/exportfs -a -r -v
exporting maple:/pub
exporting spruce:/pub
exporting maple:/home
exporting spruce:/home
exporting *:/mnt/win
```

- The -a option indicates that all directories listed in /etc/exports should be exported.
- The -r resyncs all exports with the current /etc/exports file.
- The -v option says to print verbose output. In this example, the /pub and /home directories from the local server are immediately available for mounting by those client computers that are named (maple and spruce). The /mnt/win directory is available to all client computers.

Securing Your NFS Server

Some of issues included:

- Remote root users
- Unencrypted communications
- User mapping
- Filesystem structure exposed

Opening your firewall for NFS


```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 20048 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 20048 -j ACCEPT
```

```
RQUOTAD_PORT=49001
LOCKD_TCPPORT=49002
LOCKD_UDPPORT=49003
MOUNTD_PORT=49004
STATD_PORT=49005
STATD_OUTGOING_PORT=49006
```

```
tcp   0   0 0.0.0.0:49001    0.0.0.0:*     LISTEN     4682/rpc.rquotad
tcp   0   0 0.0.0.0:49002    0.0.0.0:*     LISTEN     -
tcp   0   0 0.0.0.0:49004    0.0.0.0:*     LISTEN     4698/rpc.mountd
tcp   0   0 :::49002         :::*          LISTEN     -
tcp   0   0 :::49004         :::*          LISTEN     4698/rpc.mountd
udp   0   0 0.0.0.0:49001    0.0.0.0:*                4682/rpc.rquotad
udp   0   0 0.0.0.0:49003    0.0.0.0:*                -
udp   0   0 0.0.0.0:49004    0.0.0.0:*                4698/rpc.mountd
udp   0   0 :::49003         :::*                     -
udp   0   0 :::49004         :::*                     4698/rpc.mountd
```

Linux for Devices

Module-4, Lecture-8

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

- Parallel Virtual File System (PVFS)
- Coda
- Red Hat Global File System (GFS and GFS 2)
  - GFS 2 Packages (Fedora Core 6 and On)
  - GFS 2 Service Scripts
  - Implementing a GFS 2 File System
  - GFS Tools

- GFS File System Operations
- GFS 1

Linux for Devices

Module-4, Lecture-9

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

- Parallel Virtual File System (PVFS)
- Coda
- Red Hat Global File System (GFS and GFS 2)
  - GFS 2 Packages (Fedora Core 6 and On)
  - GFS 2 Service Scripts
  - Implementing a GFS 2 File System
  - GFS Tools

- GFS File System Operations
- GFS 1