HP World/Interex 2002

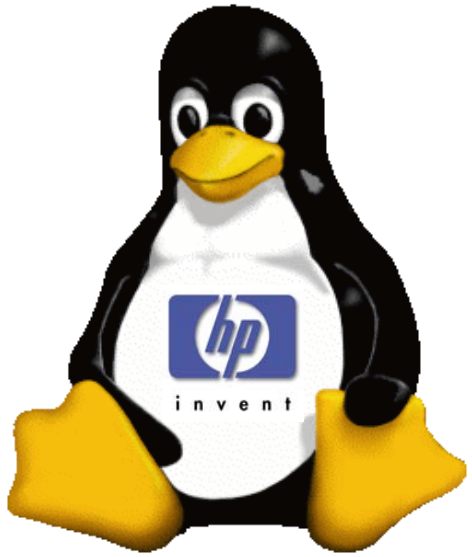
Linux System Maintenance Basics

Chris Cooper
(734) 805-2172
chris_cooper@hp.com

George Vish II
(404) 648-6403
george_vish@hp.com

hp education services
education.hp.com

Software Package Management RPM and DPKG



Version A.00

U2794S Module 12 Slides

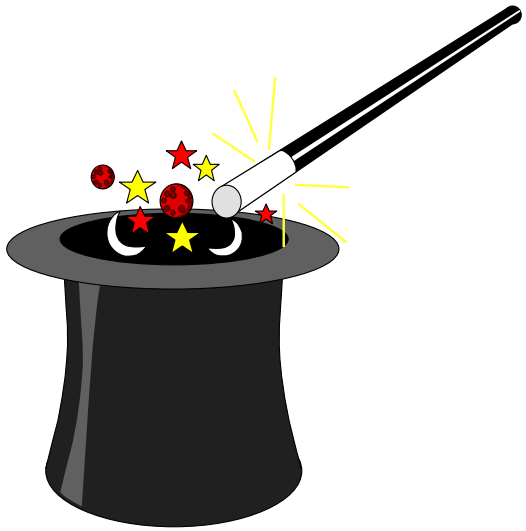
The Software Package Concept

Steps for loading software:

- Manually load and relocate files and directories.
- Extract file structure from an archive.
- Relocate and configure files by hand.
- Run any provided scripts or utilities.
- Configure start-up scripts.

OR

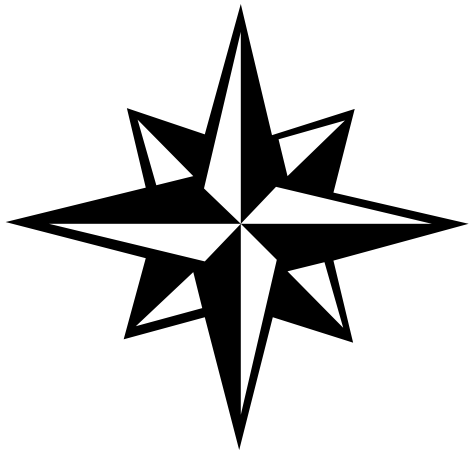
- Use a package manager to install the software.



Of Packages and Kings

Package management is a method for storing a set of related files and directories in a single archive file, along with installation scripts. When the "package" is installed, its files will be relocated on the target system, the scripts run. There have been several package managers developed to date:

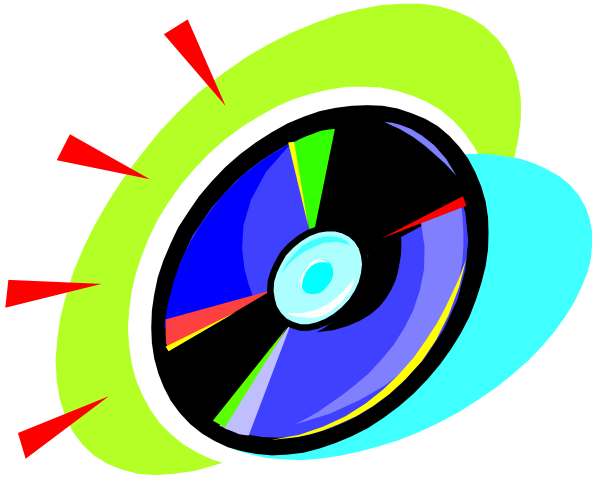
- RedHat Package Manager
- Debian Package Manager
- Slackware Package Manager
- Stampede Package Manager



Package Management

The role of the package manager is to provide a consistent means to work with packaged software. Packages can be created as either a binary or a source package. The key actions of a package manager are to:

- Install
- Remove
- Upgrade | Freshen
- Query | List
- Verify
- Build



The above actions must be performed while dealing with package-to-package dependencies, hardware, disk space, and security requirements.

Once packaged, software can easily be transported and loaded onto target systems.

Install and Remove

- Install — Load a new package:

  # rpm -i *name.version-release.architecture.rpm*

  name name of software package

  version software version number, such as 2.1.13 or 1.0

  release package release number

  architecture defines the hardware platform on which the bits will function

- Upgrade, Update an existing, or install a new package:

  # rpm -U name.version-release.architecture.rpm

- Freshen, Update an existing package only:

  # rpm -F name.version-release.architecture.rpm

- Remove, Remove an installed package:

  # rpm -e name

Query and Verify

- Query – list the packages currently present on the system:

```
# rpm -qi <package-name>
```

- Verify – examine the current status of installed packages and compare them to the original installation

```
# rpm -V <package-selection-option>
```

Rebuilding a Package

Binary packages can be rebuilt from source packages. The following command builds a new binary package, and stores it in the **/usr/src/redhat/RPMS/i386/** directory:

```
# rpm --rebuild name.version-release.architecture.src.rpm
```

The following command will compile the code, and install the named package:

```
# rpm --recompile name.version-release.architecture.src.rpm
```

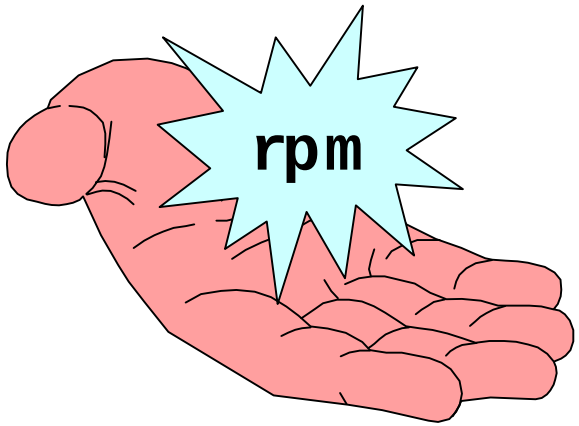
Additional Tools and Utilities

- GnoRPM
- AutoROM
- Glitter
- Apt
- Gnome Apt
- Kpackage
- Alien

Debian Equivalent Commands

The **rpm** command line

```
# rpm -i name.rpm
# rpm -U name.rpm
# rpm -e name.rpm
# rpm -qi name.rpm
# rpm -i name.src.rpm
# rpm -bb <spec-file>
```



The **dpkg** command line

```
# dpkg -i name.deb
# dpkg -i name.deb
# dpkg -r name.deb
# dpkg -print-avail name.deb
# dpkg-source -x name.dsc
# dpkg-deb -b <src dir> \
   --build <target dir>
```

hp education services
education.hp.com

Securing Your Linux System

Version A.00

U2794S Module 19 Slides

Pluggable Authentication Modules

- PAM is a set of library programs used to authenticate users.
- It provides for:
  - Authentication, such as asking for a password
  - Account checking — account is still valid, login time ok ...
  - Password setting and changing
  - Session checking when user is authenticated
    - Are certain actions allowed for this user?
- Functions are implemented using library modules.
  - Can be stacked
- PAMs are controlled and configured for individual programs through the **/etc/pam.d** configuration directory.
  - Every controlled service is identified by a file. The file name is the program name.

PAM Service File

- A PAM service file contains configuration information for a restricted service.
- The file contains three fields.
- In some files, you may find a fourth field for optional arguments.
- The fields are:
  - Module type
  - Control flag
  - Module path
  - Arguments

Module Type

- The PAM standard defines four types of modules:
  - **The auth** type is establishes that users are who they claim to be.
  - **The account** type checks to see if a user is allowed to use a service.
  - **The password** type updates authentication tokens.
  - **The session** type performs certain actions when the user logs in or logs out.

Control Flag

- **Requisite**: if a service returns a failure, no other modules will be evaluated.
- **Required**: if a particular module fails, other modules of the same type will still execute.
- **Sufficient**: if no other module of this type has failed, then the success of the module is sufficient to guarantee that security requirements have been met.
- **Optional**: PAM ignores the module failure and continues processing the next module in sequence.

Module Path

- The compiled-in modules are located in the **/lib/security** directory.
- Modules interface with the file or device that gives the SUCCESS, FAILURE, or ignore to authentication.
- PAM then passes either SUCCESS or FAILURE to the service requesting the information.
- Modules can be stacked, requiring more than one type of authentication, or allowing any of several types.

Optional Arguments

- Some modules can receive specific options. It is up to the module to parse and interpret the options.
- This field can be used by the module to turn on debugging or to pass any module-specific parameters, such as a timeout value.
- System administrators can use the option field to fine tune the PAM modules.

Examples

- **Example 1: Blocking Anyone to `su` to `root` except members of `wheel`**

  Add the following two lines to the top of the **`su`** file:

```
auth sufficient /lib/security/pam_wheel.so trust use_uid
auth required /lib/security/pam_wheel.so use_uid
```

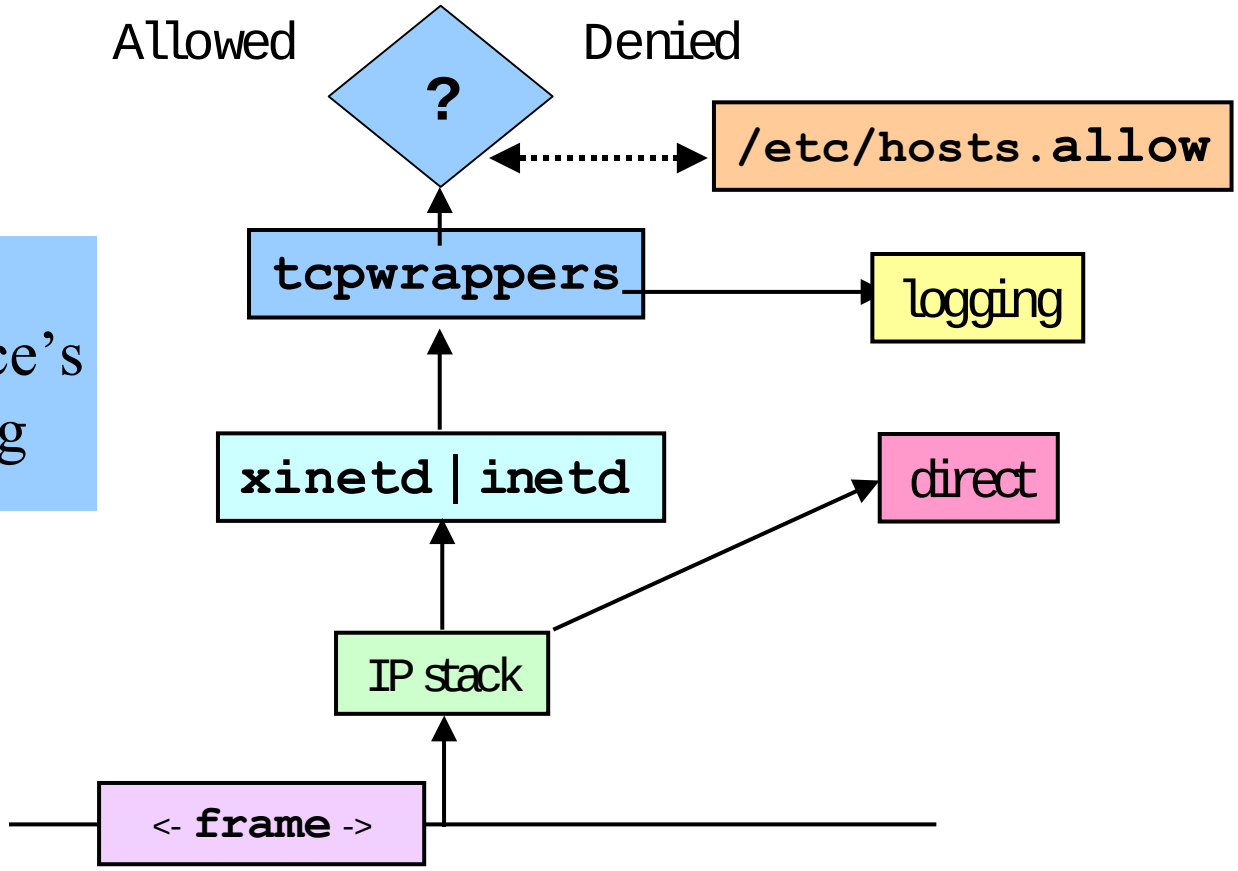
- **Example 2: Setting Linux System Resource Limits**

  Add the following line to the bottom of the **`login`** file and then edit the `/etc/security/limits.conf` file:

```
session required /lib/security/pam_limits.so
```

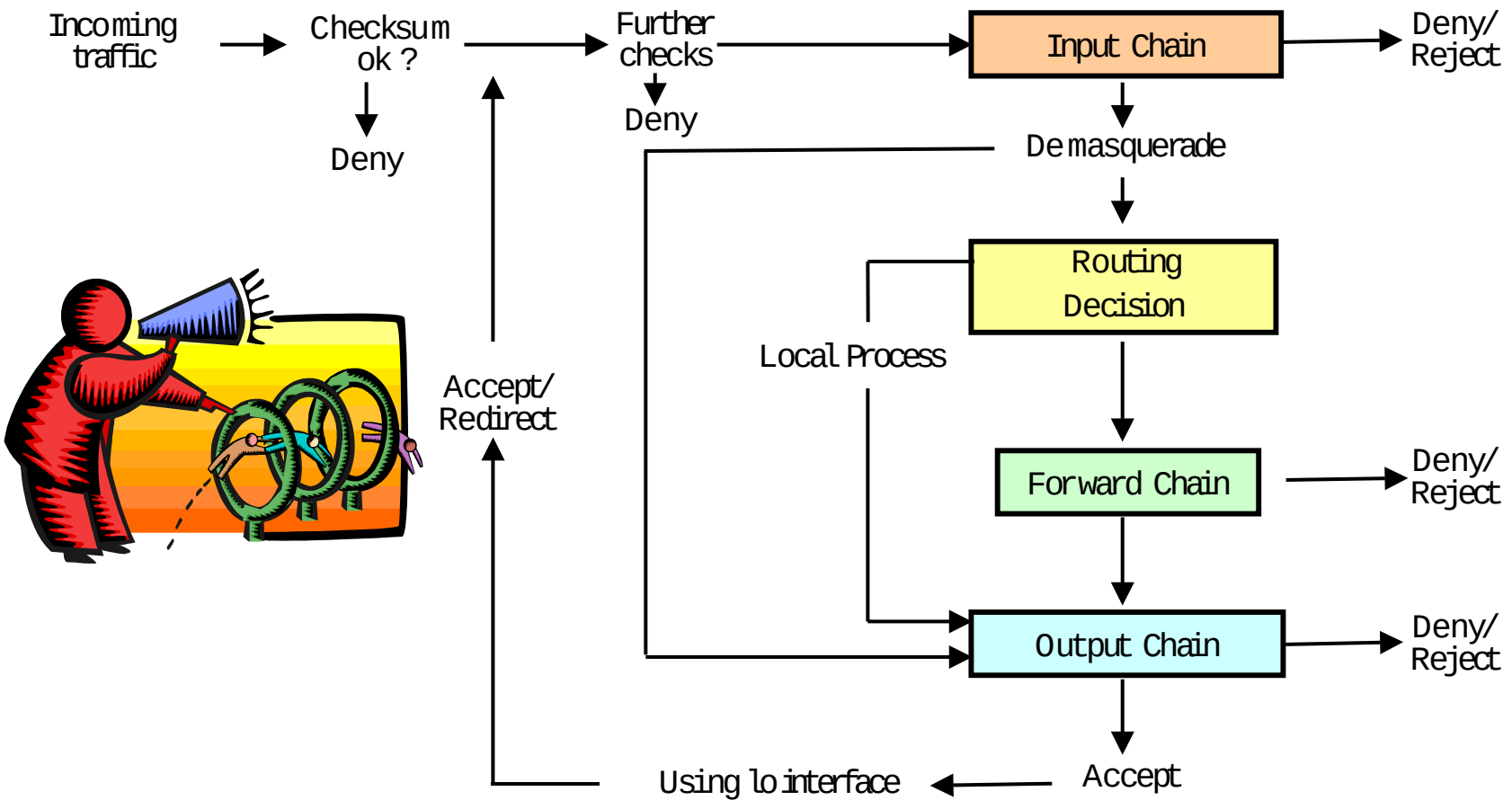
TCP Wrappers

Provides protection around TCP/IP service's access control logging



Linux's ipchains - iptables

IPCHAINS operation:

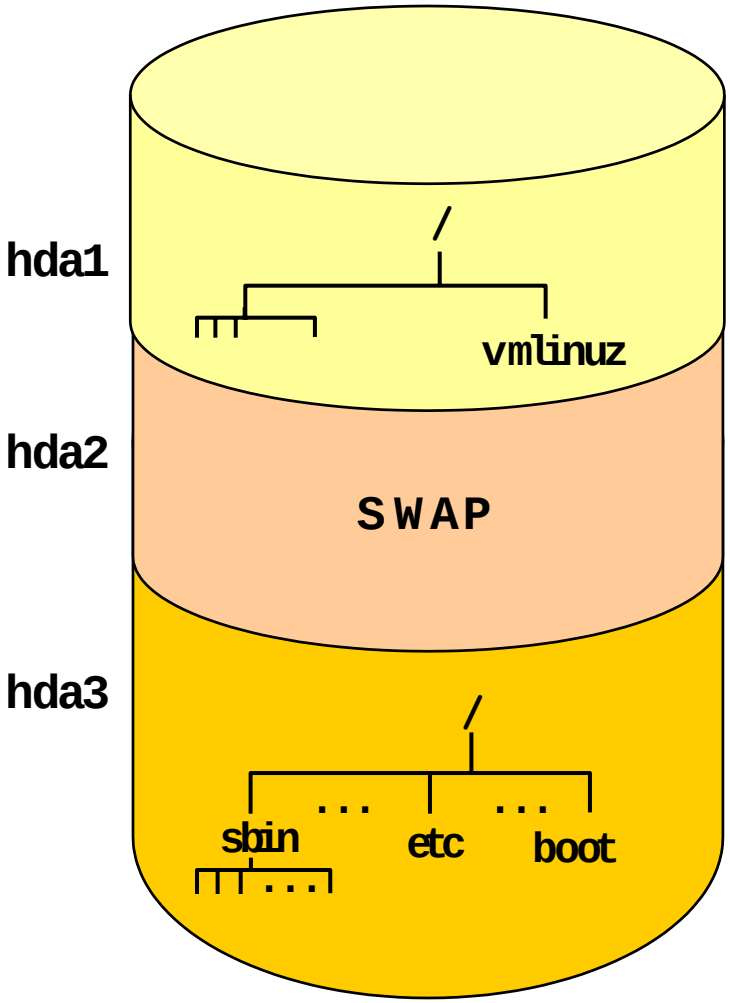


Recovering a Non-Bootable Linux System

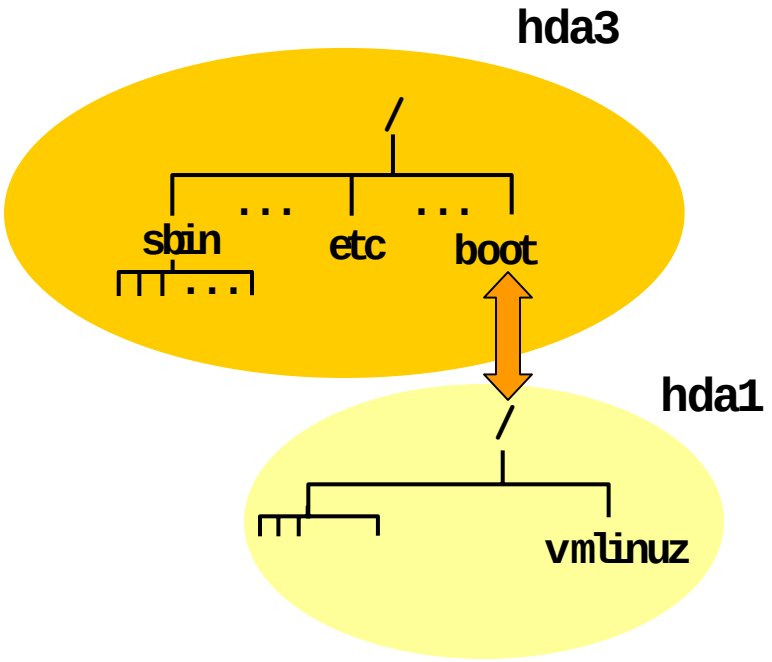
Version A.00

U2794S Module 20 Slides

Normal Boot Process



Physical File/Disk Structure



Logical File/Disk Structure after boot

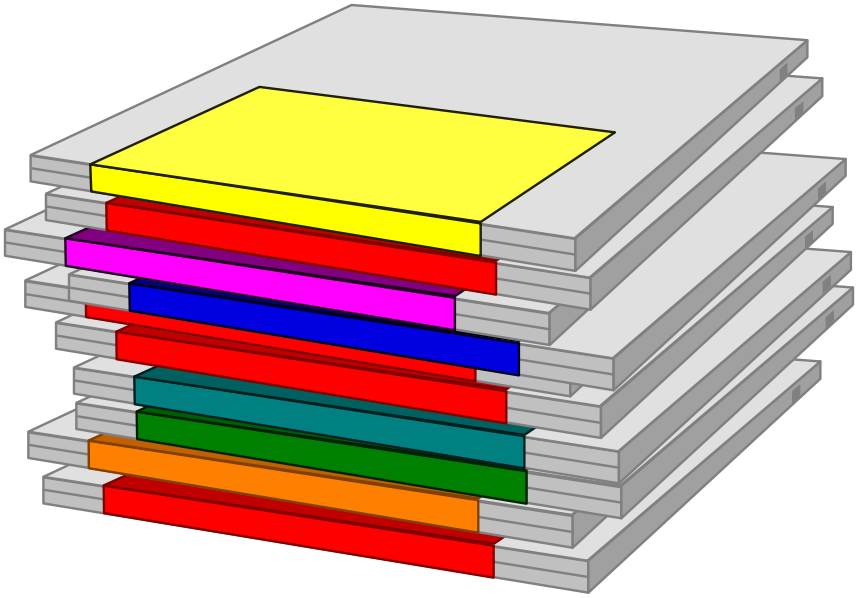
Kernel Corrupt or Missing

- The kernel is usually named **/boot/vmlinuz** (compressed).
  - If the kernel is uncompressed, its name is **/boot/vmlinux**.
- If the kernel is deleted by accident, boot from the rescue floppy created at installation time.
- After boot, mount **/boot** to a temporary location.

```
# mount /dev/hda2 /mnt/broken_boot_fs
```

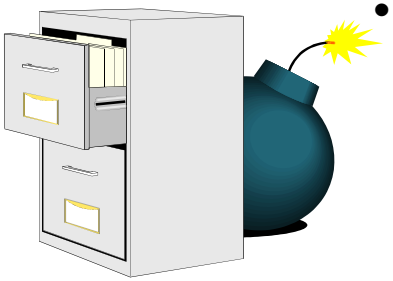
- Copy the kernel from the rescue floppy to **/boot/vmlinuz** (or other appropriate kernel name).
- Reboot with the restored kernel.
- Configure and recompile a new kernel, if necessary.

Third-Party Rescue Disks



- The rescue floppy won't work if important parts of the system (like **/etc** or **/usr**) are damaged.
- For more extensive repairs, you need a "tiny Linux distribution" that is customized for this purpose.
- These "repair kits" come with tools like **fsck** and network support.
- Examples:
  - MuLinux
  - Trinux
  - Tom's Root Boot
- Some recent distributions provide "rescue disk" capabilities on their installation CD (ie. Red Hat 7.2)

Back Up Your Data!



- Backups are critical for the operation of your system, because of:
  - user error, administrative errors, hardware failure, software failure, destructive break-ins, theft, disaster, and archival needs.
- Backup:
  - user files, log files, important system directories, your RPM database
- Use consistent backup strategy:
  - Weekly or monthly full backup
  - Daily partial backup cycle
- Very common to use **tar**
  - Commercial products are available

Backup Strategies for /boot

- Backups are important!
  - Use reliable and supported media.
  - Schedule backups regularly and DO THEM!

- Backing up with tar:

```
# tar --create --file /dev/st0 /usr/boot
```

or `# tar -cf /dev/st0 /usr/boot`

```
# tar -cMf /dev/fd0H1440 /usr/boot
```

(for multiple floppy disk volumes)

```
# tar --compare  --verbose -f /dev/st0 /dev/boot
```

or `tar -cvf /dev/st0 /dev/boot`

(to compare the backup against the original data)

Restoring Files with **tar**

- Extract files from a tar backup archive:

```
# tar --extract --same-permissions --verbose --file /dev/st0
```

or `# tar -xpvf /dev/st0`

- List files in an archive:

```
# tar --list --file /dev/st0
```

or `# tar -lf /dev/st0`

- Extract specific files from an archive:

```
# tar xpvf /dev/st0 \
usr/src/linux-1.2.10-includes/include/linux/hdreg.h \
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
```