hp education services
education.hp.com

HP World/Interex 2002

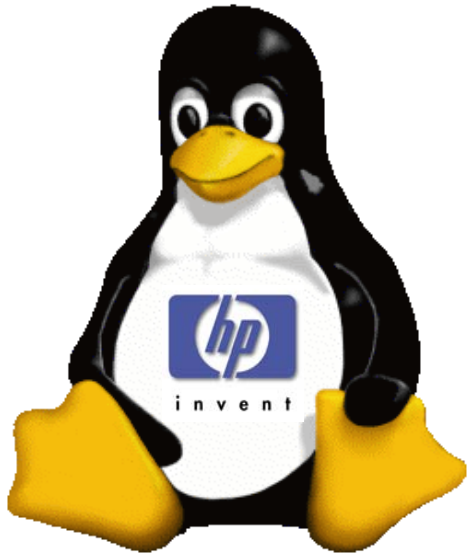
Linux File System Support

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

Linux File System Management



Version A.00

U2794S Module 11 Slides

Why Use Different File Systems?

- Linux supports multiple file system types.
- File systems types include: disk-based, CD-ROM and network-based file systems.
- Advantages of this support include:
  - Data can be made available when required.
  - Old and new file system types can be used together.
  - Backups can be taken offline while other file systems are in use.
- Disadvantages include:
  - Administration overhead increases.
  - Backup policy becomes more complicated.

File Systems Supported under Linux

- The Linux operating system supports many types of file systems.
- File systems are created in disk partitions. (or Logical Volumes)
- The following are a few of the file system types supported under the Linux Kernel

- ext2fs
- ext3fs
- reiser
- JFS
- Veritas
- NFS
- RFS
- SMB (CIFS)
- Minix
- Vfat
- Fat16
- Fat32
- MSDOS
- NTFS
- ISO 9660
- And many, many more!

The **ext2fs/ext3fs** File Systems

- The **ext2fs** file system consists of:
  - a super block (with backup super blocks spread over the partition)
  - a cylinder group block for each cylinder group
  - an inode table (spread over the available cylinder groups)

  (This is the file system that is most similar to the classic UFS)

- The **mkfs (mkfs.ext2)** command is used to create the file system and the **fsck (fsck.ext2)** command is used to check it's consistency and make repairs.

- The recently released **ext3fs** (available on 2.4.x kernel distributions) adds the feature of a recovery journal to the basic layout of the **ext2fs**. Early experimentation shows it to be reliable.

ext2fs File System Format

- The **ext2** file system consists of a number of cylinder groups.
- Each cylinder group contains a portion of the inode table.
- Data is written, whenever possible, in contiguous data chunks.
- Disk fragmentation is kept to a minimum (usually less than 3%).

/dev/hda1				
CylGrp1	CylGrp2	CylGrp3	CylGrp4	CylGrp5
IIxxddddddxx	IIxxxddddddx	IIxxxxdddddd	IIxddd	IIxxx

ext2fs Super Block

- The super block contains the following details:
  - size of the partition
  - size of remaining space within the partition
  - number of cylinder groups
  - size of cylinder groups
  - total number of inodes
  - number of free inodes
- Backup super blocks are spread over the disk, one per cylinder group.

Journal File Systems for Linux

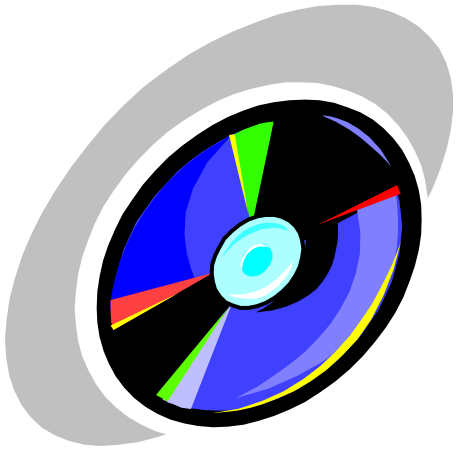
- Much recent development has been centered on providing Linux with fast recovery Journal File Systems.
- The Rieser File System has been in development for some time and has been released for evaluation.
- IBM recently "opened" their JFS file system and the Linux port has been proceeding rapidly.
- It seems to be feast of famine, prior to the 2.4.x release there was no available file systems with the journal feature and now we have at a minimum three open source choices !

FAT16/FAT32/Vfat/MSDos/NTFS

- These file systems are used by the Microsoft Windows operating systems (95/98/2000/NT).
- These file systems suffer from severe fragmentation over prolonged use and require regular defragmentation in order to provide reasonable performance.
- A number of software tools (**mtools**) are provided with Linux to support the FAT16/FAT32 file systems.
- While the NTFS file system may be accessed by the Linux kernel as a R/O file system, building the Linux kernel with NTFS write capabilities is considered a developmental and somewhat risky endeavor. Until it is further refined we suggest only using the R/O mount functionality.

CD-ROM File Systems

- This file system type is used for the CD-ROM file storage
- The ISO 9660 format is used by the IT and Software industry.
- Even CD's containing Micro-Soft OS file follow this standard.

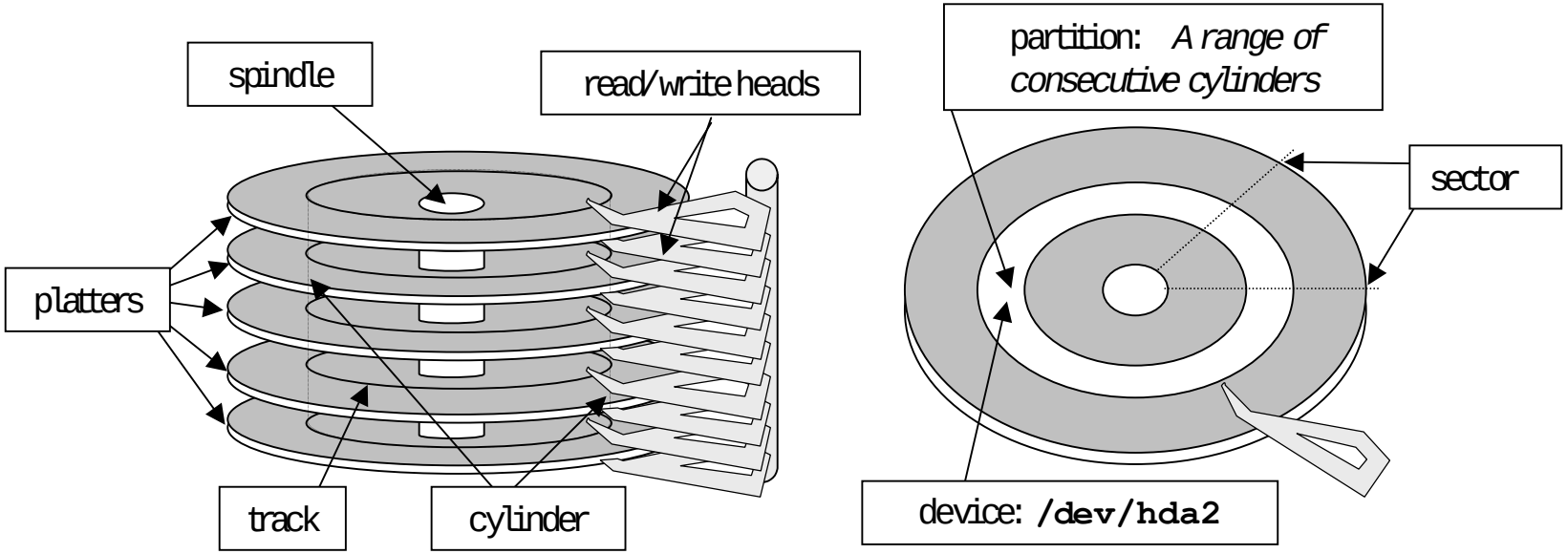


Other File Systems

- Other file systems supported by Linux include:
  - Veritas (VxFS) file system (Veritas Software Corporation)
  - Network File System (NFS) (Sun Microsystems, Inc.)
  - Remote File System (RFS)

Devices and File Systems

- A disk can be subdivided into partitions.
- Partitions can contain file systems.
- Disk partitions are accessed through a device file (for example, **`/dev/hda2`**).
- Data is written in *disk blocks*
  (historically 1 *disk block* = 1 *sector* = 512 *bytes*).



Disk Partitions

- Disk partition details can be displayed using any of the following commands:

  # fdisk -l

  # hdparm -g /dev/sd@

  where @ is the letter associated with the hard disk drive.

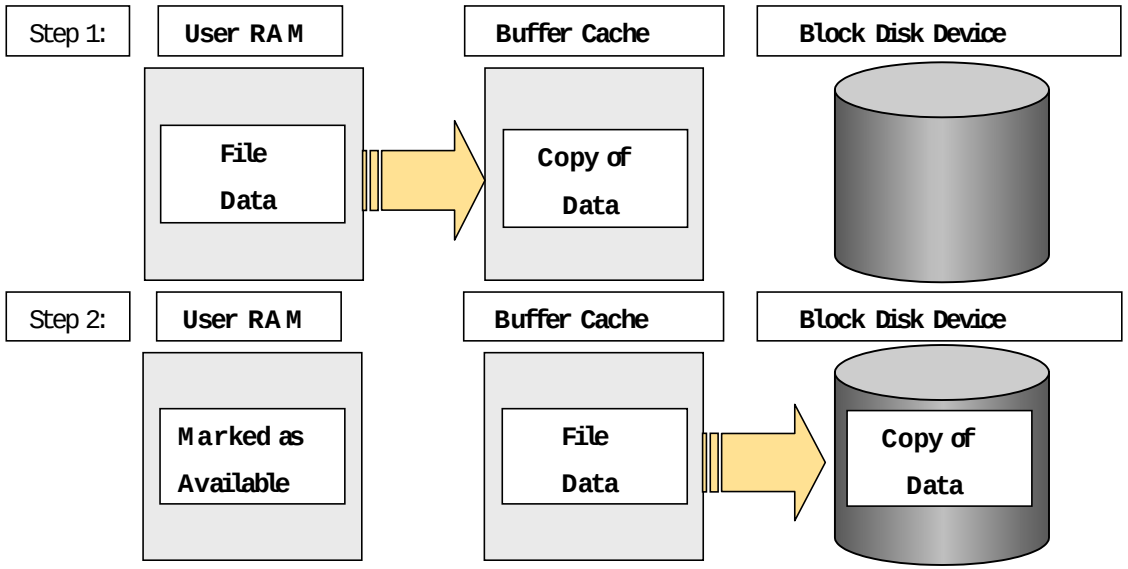
  # parted

  (at the prompt enter **print** then **quit** to exit)

- These commands display the geometry details for the disk (number of cylinders/number of tracks per cylinder/number of sectors per track).

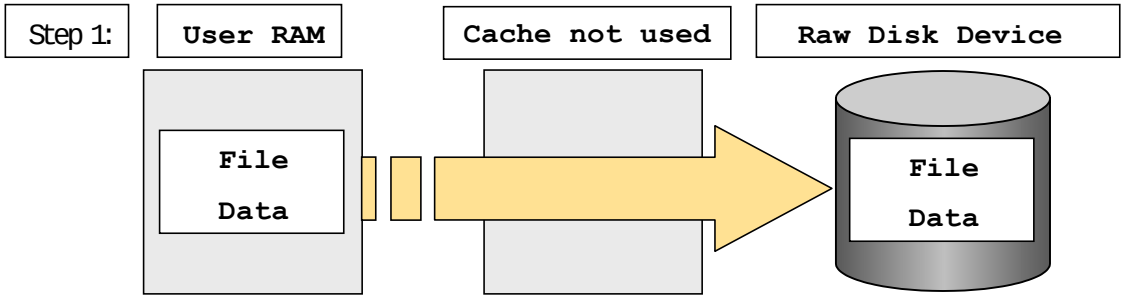
Block Disk Devices

- When using block disk devices, data is first written to a buffer cache, then written to disk using blocks.
- The data is retained in the buffer cache until that area of cache is required by another disk input/output operation.



Character (Raw) Disk Devices

Data is written as a stream of characters.
Data is not written to buffer cache.
Data is written on disk as a consecutive stream of characters.



Making a File System **mkfs**

- File systems are made using the **mkfs** command.
- Before using the **mkfs** command, the administrator must know:
  - What file system type is to be made?
  - Are there special options that must be used?
  - What is the device name of the storage medium to be used?
  - Should space be reserved for the **root** user?

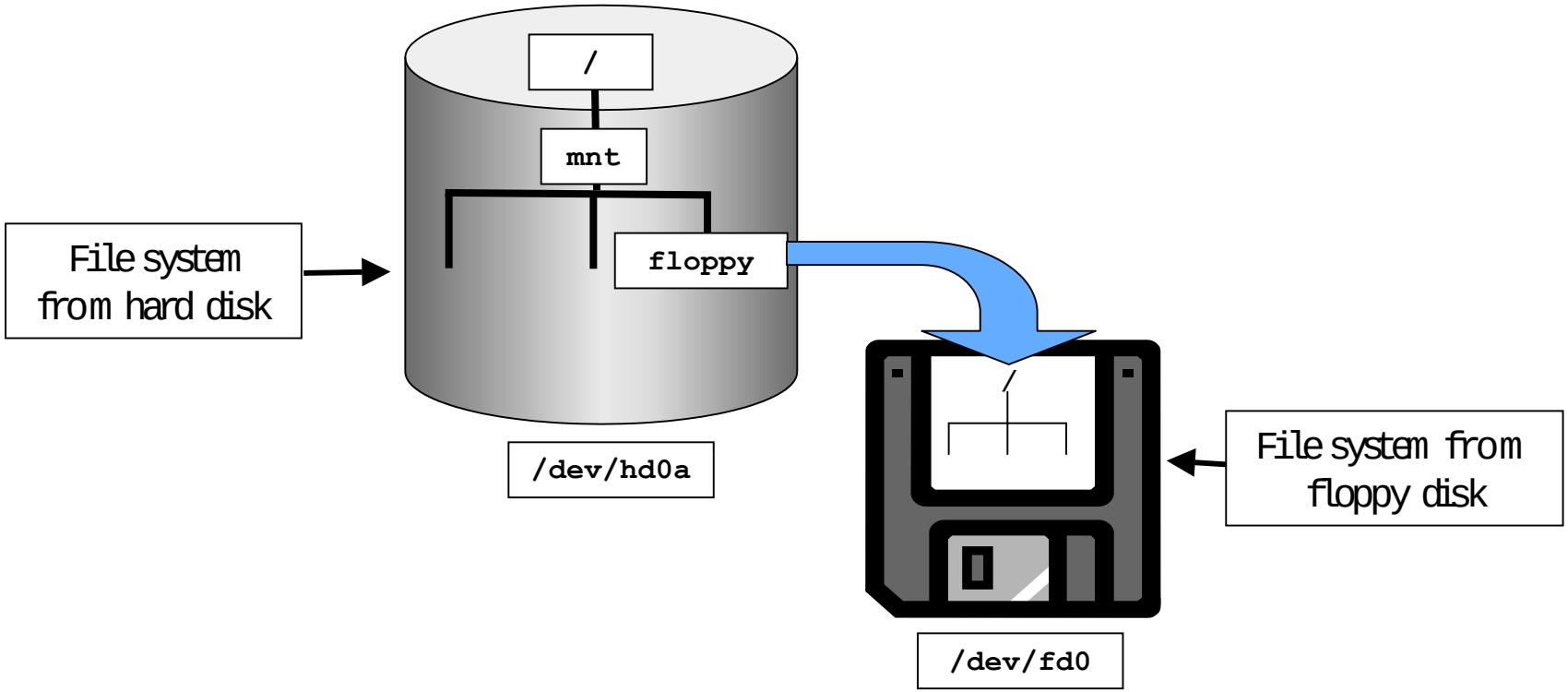
- An example **mkfs** command:

```
# mkfs  -t  ext2  /dev/sd@
```

The **mount** Command

The **mount** command is used to mount file systems to the directory hierarchy.

```
# mount /dev/fd0 -t msdos /mnt/floppy
```



Mounting at Boot Time — the **fstab** file

The **/etc/fstab** file is read by the **mount** command at boot time and subsequent to a system boot.

```
# cat /etc/fstab
```

```
/dev/hda1     /              ext2     defaults     1      1
/dev/hda5     swap           swap     defaults     0      0
/dev/fd0      /mnt/floppy    ext2     noauto       0      0
/dev/cdrom    /mnt/cdrom     iso9660  noauto,ro    0      0
none          /proc          proc     defaults     0      0
none          /dev/pts       devpts   mode=0622    0      0
```

Device to Mount	Mount-point Directory	F/S Type	Mount Options	Backup Check	fsck Pass

Adding a Hard Disk

The following steps are required for adding a hard disk drive unit (after adding the hard disk drive to the system):

1. Run **`fdisk`** to create the necessary partitions.
2. Create file system(s) in the relevant partition(s).
3. Create a mount-point directory(s).
4. Edit the **`/etc/fstab`** file to allow automated **`mount`**.

The Inode Table

- Files and directories have an inode number.
- The system maintains a lookup table by reading the directory files and storing sets of information in RAM.
- The inode table is stored on disk.
- Inodes are found in cylinder groups on disk.
- Inodes contain the ownership and permission details of the file/directory.
- Inodes contain pointers to the file/directory data on the disk.
- A file will, generally, be created in the same cylinder group area as the directory in which that file exists.
- The **mkfs** command creates the inode table.

File System Health

- The **fsck** command verifies the inode table entries.
- **fsck** is used to maintain the integrity of the file system structural data.
- It can be run at boot-time, controlled by an entry in **/etc/fstab**.
- It may be run manually by the **root** user.
- **fsck** should not be run on a mounted or in-use file system.
- The syntax for **fsck** is:

```
# fsck [-AVRTNP] [-s] [-t fstype] filesystem
```

- **fsck** should not be run on the **root** file system while in read/write mode, this could corrupt the file system.

fsck

- Several checks are run by **fsck**
  These are:
  - Checking inodes, blocks and sizes
  - Checking directory structure
  - Checking directory connectivity
  - Checking reference counts
  - Checking group summary information
- **fsck** checks the consistency of data stored in the inode table and will fix any corruption, if possible
- Files may be "recovered" in the **lost+found** directory.

Out of Inodes

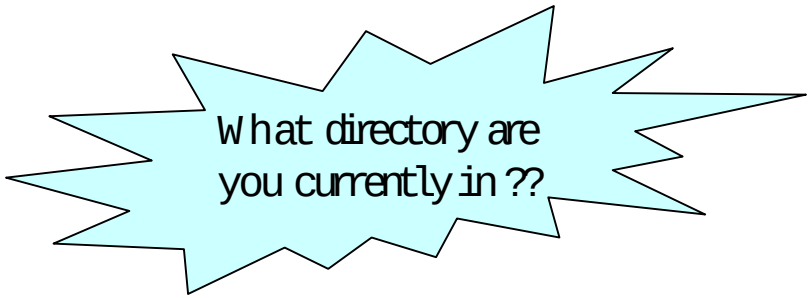
- Users view disk storage as "volume" in Megabytes.
- The available disk space can be checked using **df -k**.
- It is possible to run out of inode table entries before all of the disk space has been used up (on some file system types, ie. ext2fs).
- The remaining inode count can be checked using **df -i**.

```
# df -i
File system       Inodes  IUsed  IFree IUse% Mounted on
/dev/hda1         131616   7694 123922   6% /
/dev/hda9          66400    633  65767   1% /home
```

Problems with Unmounting Disks

- Partitions cannot be unmounted when the file system is in use.
- File systems can be in use by both system and user processes.
- If an attempt is made to unmount a file system that is in use, a warning message is displayed.

```
# umount /spare2
umount: /spare2: device is busy
#
```



fuser

- The **fuser** command can be used to show which processes are using the file system.
- The **-k** option can be used to "**kill**" the process(es) that is/are using the file system.
  - **# fuser /spare2**
  - **/spare2: 1010c**
  - **# fuser -u /spare2**
  - **/spare2: 1010c(root)**
  - **# fuser -ku /spare2**
  - **/spare2: 1010c(root)**

The "c" suffix denotes that associated pid# has a current directory open, "e" is an executable, "f" is an open file, "r" is a root directory, "m" is a mapped file

Additional File System Controls

Disk Quotas:

- Enable quotas to limit users taking up too much disk space.
- Configure **/home** to be on a separate partition.
- To enable quotas:
  - Change entry in **/etc/fstab**, add **usrquota** and/or **grpquota** to the options.
- Remount the partition with the **mount** command..
  - Use **quotacheck** to calculate existing usage.
  - Use **quotaed** and **quotaon** to configure and switch on quota management.
- Test quota management for your users.
- Let your users know you have enabled quotas.

Immutable Files:

- The **ext2** filesystem has additional attributes. We will focus on the **I** attribute. The **I** attribute makes a file immutable. When set, the file cannot be renamed, modified, deleted, or even linked to.
- Only the system administrator can set this attribute.
- The **I** attribute will not allow even **root** to do any modifications to the file.
- Use the **chattr** command to set the file's attribute.
- Use the **lsattr** command to show the file's attribute.
- Example:

```
chattr +i /etc/inetd.conf
```