# System Administration

## Filesystems

UNIX

# Filesystems

- **managing the UNIX filesystem is one of the most important tasks**
  - can get rather complex in a heterogeneous, distributed environment
- **Unix supports a wide variety of different filesystems**
  - *(see p.412, Frisch)*
  - note the confusion in naming
    - *typical of the UNIX world*

# Adding A Disk

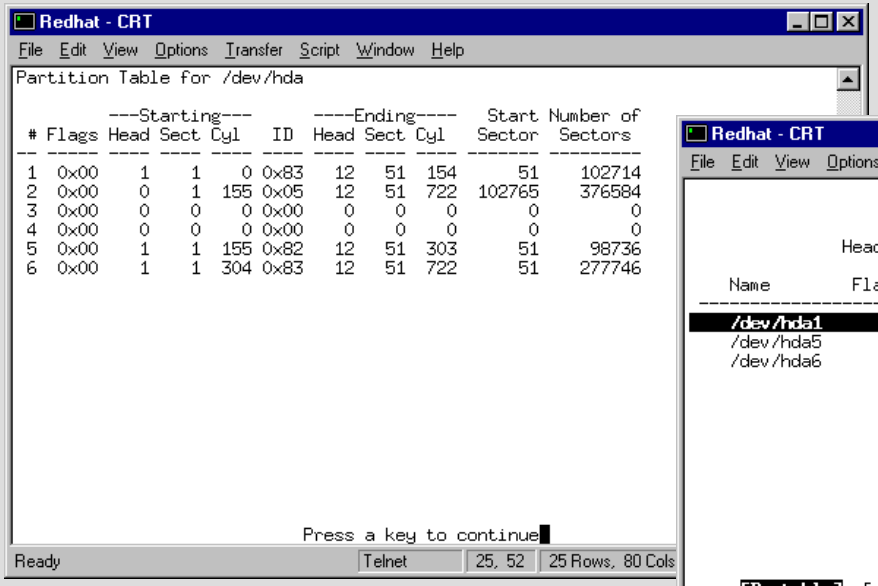
- **adding a disk requires several steps**
  - create a kernel with suitable driver support
  - make the special devices for the disk
  - physically attach the disk
  - partition the disk
  - make a filesystem on the disk
    - *"formatting a disk under DOS is equivalent to making a filesystem under UNIX…UNIX disk formatting is equivalent to what DOS calls a* low-level format*."*
  - update the filesystem configuration table, /etc/fstab

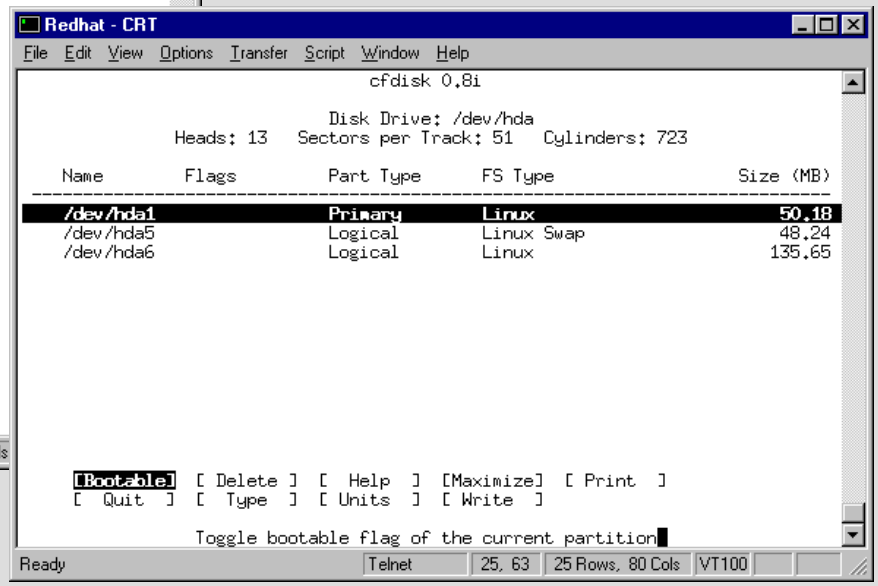
```
/dev/hda1          /          ext2    defaults        1 1
/dev/sdb1          /home      ext2    defaults        1 2
/dev/sdb5          /usr       ext2    defaults        1 2
…
```

  - mount the filesystem into the directory tree
  - prepare for use

# Partitioning A Disk

- typically a vendor/machine/revision level specific activity
  - linux: fdisk, cfdisk, disk_druid
  - sun: format; SCO: divvy; IRIX: fx
  - may need to be done from the boot monitor, rather than from within UNIX



# Partitioning Schemes

- **why partition?**
  - increase reliability
    - *a crash may only kill part of the disk, the other partitions may remain OK*
  - increase utilization of disk space
    - *small partition means smaller blocks: less fragmentation*
  - overcome restrictions
    - *lilo can't boot from a file contained above cylinder 1023*
- **there may exist a convention for partitioning disks for a given version of UNIX**
  - e.g. BSD, *(see p.410, Frisch)*
    - partition c in this scheme often used for low-level, whole-of-disk tasks like backups, checks, etc.
  - convention can be ignored if required
  - linux uses DOS' idea of primary, extended and logical partitions
    - *linux only allows 4 real partitions*
      - partitions 1-4 are real
      - partitions 5+ are logical, contained within an extended partition

# Making A Filesystem

- mkfs command

```
# mkfs -v -t ext2 /dev/sda1
mke2fs 1.10, 24-Apr-97 for EXT2 FS 0.5b, 95/08/09
mkfs.ext2: bad blocks count - /dev/sda1
```

- don't *always* have to make a filesystem
  - some databases use a raw partition and prepare it specially
    - *for speed or to support special features*
  - swap partition is also typically unformatted
    - *also for speed*

```
# dumpe2fs /dev/sda1
dumpe2fs 1.10, 24-Apr-97 for EXT2 FS 0.5b, 95/08/09
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          28250484-558d-11d2-96a3-d9e46f6b64e6
Filesystem magic number:  0xEF53
Filesystem revision #:    0 (original)
Filesystem features:      (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              5136
Block count:              20464
Reserved block count:     1023
Free blocks:              19796
Free inodes:              5125
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1712
Inode blocks per group:   214
Last mount time:          Sun Nov 22 22:15:27 1998
Last write time:          Fri Oct 30 16:58:14 1998
Mount count:              16
Maximum mount count:      20
Last checked:             Fri Oct 30 16:58:14 1998
Check interval:           15552000 (6 months)
Next check after:         Wed Apr 28 16:58:14 1999
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)

Group 0: (Blocks 1 -- 8192)
  Block bitmap at 3 (+2), Inode bitmap at 4 (+3)
  Inode table at 5 (+4)
  7961 free blocks, 1701 free inodes, 2 directories
  Free blocks: 232-8192
  Free inodes: 12-1712
Group 1: (Blocks 8193 -- 16384)
  Block bitmap at 8195 (+2), Inode bitmap at 8196 (+3)
  Inode table at 8197 (+4)
  7974 free blocks, 1712 free inodes, 0 directories
  Free blocks: 8411-16384
  Free inodes: 1713-3424
Group 2: (Blocks 16385 -- 20463)
  Block bitmap at 16387 (+2), Inode bitmap at 16388 (+3)
  Inode table at 16389 (+4)
  3861 free blocks, 1712 free inodes, 0 directories
  Free blocks: 16603-20463
  Free inodes: 3425-5136
```

# (Un)Mounting A Filesystem

- filesystems are linked into the directory heirarchy
  - mount

```
# mkdir /documentation
# mount -o ro /dev/sda1 /documentation
```

  - at a specified *mount-point*
    - *a directory that already exists*
      - if the directory is not empty, its contents are 'hidden'
  - umount command removes the filesystem from the directory tree

```
# umount /documentation
# rmdir /documentation
```

- mount -a -t nonfs,proc used at boot time
  - looks in the file /etc/fstab
- fuser
  - shows what processes are using a given filesystem
    - *makes it possible to know when a filesystem can be unmounted*

```
# fuser -u /
/:                    350r(bob)    416r(bob)
```

- look at nfs filesystems and the automounter later

# FSSTND/FHS

- **a standardized filesystem layout**
  - *http://www.pathname.com/fhs*
  - originated with Linux but being adopted by other manufacturers
- **standardization should cure some problems:**
  - /bin, /sbin and /usr/bin don't always have well-defined divisions: the distribution of the binaries between these directories varies greatly between systems.
  - having both binaries and configuration files in /etc makes this directory more confusing and more difficult to maintain
  - many common implementations of /usr cannot be mounted read-only because they contain variable files and directories that need to be written to
  - in a networked environment it is desirable to serve software to workstations via NFS. Such filesystems should ideally be mounted read-only so that accidents or malice on one workstation cannot damage the files on the server.

# More FSSTND/FHS

- top-level directories
  - /bin: the basic executables; used at both boot-time and in a running system
  - /boot: kernel and other boot-time binaries
  - /dev: system devices
  - /etc: system-specific configuration files
  - /lib: libraries needed to execute the binaries in /bin and /sbin
  - /mnt: mount point for transient devices
    - *cdrom, floppy, zip drive, etc.*
  - /proc: process filesystem mount point
  - /sbin: files essential for booting the system
  - /tmp: temporary working space
    - *often a separate partition for speed*
  - /usr: files that are shareable across a whole site
    - *should be mountable read-only*
    - *from CD-ROM or via NFS*

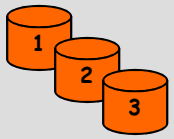
```
/
|
+--bin
+--boot
+--dev
+--etc
+--home
+--lib
+--mnt
+--proc
+--root
+--sbin
+--tmp
+--usr
| \
| +--X11R6
| +--bin
| +--lib
| +--local
| | \
| | +bin
| | +lib
| +--src
|
+--var
  \
  +--adm
  +--lib
  +--lock
  +--log
  +--preserve
  +--run
  +--spool
  | \
  | +--mail
  | +--mqueue
  | +--news
  | +--smail
  | +--uucp
  +--tmp
```

# Even More FSSTND/FHS

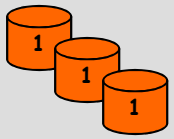
- **other directories**
  - /home: user's home directories
    - *but root's home directory is in /root on the root partition*
    - *accessible even if no other filesystems are mounted*
  - /var: variable data files
    - *logs, spool files etc.*
    - *putting variable data here means that /usr can be read-only*
  - /usr/man: sources for the on-line documentation
  - /usr/local: for use by the system administrator when installing software locally. Needs to be safe from being overwritten when the system software is updated
    - *may have another purpose*
      - *"Since system upgrades from Red Hat Software are done safely with the RPM system and Glint, …, we recommend you use /usr/local for software that is local to your machine."*
  - /opt : add-on application software packages
  - /usr/games
  - /usr/X11R6: X Window stuff
  - /usr/src: UNIX source code

# RAID

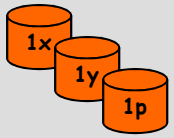
- using multiple disks for speed/reliability/recoverability
  - various configurations
    - *RAID 0: STRIPING*




      - combine partitions into one logical device in such a fashion as to fill them up evenly, one chunk here and one chunk there. Increases throughput if the partitions reside on distinct physical disks.
    - *RAID 1: MIRRORING*




      - creates partitions which are exact copies of each other. In the event of a mirror failure, the RAID driver will continue to use the remaining partitions in the set, providing an error free device. In a set with N drives, the available space is the capacity of a single drive, and the set protects against a failure of (N - 1) drives.
    - *RAID 4/5: PARITY*




      - a RAID set of N drives with a capacity of C MB per drive provides the capacity of C * (N - 1) drives, and protects against a failure of a single drive. For a given sector, (N - 1) drives contain data sectors, and one drive contains the parity protection. For RAID-4, the parity blocks are present on a single drive, while RAID-5 distributes the parity across the drives. The parity blocks allow all the data to be reconstructed if a drive fails.
  - RAID support needs to be built into the kernel
  - *(see p.441, Frisch)*

# Logical Volumes

- **available on Digital UNIX, HP/UX, IRIX and AIX**
  - as the *Logical Volume Manager*
  - Digital UNIX now provides the *Logical Storage Manager*
    - *layered on (& replacing) the LVM*
- **amalgamating physical disks as a single logical device**
  - *(see p.456, Frisch)*
- **benefits**
  - filesystems and files can be larger than a physical disk
  - filesystems can be dynamically resized
    - *usually only increased*
  - mirroring and striping is typically supported
    - *software RAID*
- **Digital UNIX LVM example**
  - create a volume group named /dev/my_vg from three physical volumes; create a logical volume on /dev/my_vg called my_lv of 50 4Mb extents (the default) in size:

```
# vgcreate /dev/my_vg  /dev/rz3c  /dev/rz5c  /dev/rz6c
# lvcreate -n my_lv -l 50 /dev/my_vg
# newfs /dev/my_vg/rmv_lv
# mount /dev/my_vg/mv_lv /my_lv
```

# Journalled Filesystems; Vold

- **AIX, HP/UX and IRIX provide *journalled filesystems***
  - for extra filesystem reliability
  - *filesystem* changes are logged to a dedicated disk area
    - ***file** changes are not logged*
  - effectively, all filesystem manipulations are turned into logged transactions
  - if system crashes, it is possible to reconstruct it by 'replaying' recent filesystem changes
- **vold**
  - volume management daemon used to manage CD-ROM and floppy devices
    - *found on Sun systems*
      - configured via /etc/vold.conf
    - *creates and maintains a rooted file system image that contains symbolic names for floppies and CD-ROMs*
      - can be used to do things such as autoplay an audio CD or automount a data CD-ROM

# Other Filesystem Stuff

- **badblocks**
  - search for unusable disk blocks on a device
- **tune2fs**
  - allow modification of filesystem parameters
    - *developer tool: never really used*
- **fsdb**
  - filesystem editor—only on some systems
- **usermount**
  - 'friendly' RedHat GUI tool for (un)mounting filesystems
- **sync**
  - forces all pending writes so that memory and disk are the same
- **du, df**
- **umask**
  - used to set initial file permissions on a newly-created file
- **quotas**
  - see later

# Dealing With DOS

- **mtools**
  - simple support for DOS disks
    - *ported to many UNIX platforms*
    - *mformat, mdir, mattrib, mcopy, mdel, etc.*

```
% mcopy "a:*.txt" .
```

- **native Linux support**
  - Linux can mount DOS formatted disks just like any other filesystem

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