Filesystems and Disks

Managing the UNIX filesystem is one of the SA's most important tasks.

- Making local and remote files available to users
- Monitoring and managing the system's finite disk resources
- Protecting against file corruption, headware failures, and user errors via a well-planned backup schedule
- Ensuing data confidentiality by limiting file and system access
- Checking for and correcting file system corruption
- Connecting and configuring new storage devices when needed

Filesystem Types

Use	AIX	FreeBSD	HP-UX	Linux	Solaris	Tru64
Default local	Jfs or jfs2	Ufs	vxfs	Vxfs	Ufs	Ufs or advfs
NFS	nfs	nfs	nfs	nfs	nfs	Nfs
CD-ROM	cdrfs	Cd9660	cdfs	Iso9660	hsfs	Cdfs
swap	Not needed	swap	Swap, swapfs	swap	swap	Not needed
/proc	procfs	procfs	Not supported	procfs	procfs	Procs
RAM-based		mfs		Ramfs,tmpfs	tmpfs	mfs

Mounting and umounting filesystems

Mounting is the process that makes a disk's content available to the system, merging it into the system directory tree

Use the **mount** command to mount a filesystem
Use the **umount** command to umount a filesystem

- Manually mount a file system
  # mount block-special-file mount-point

  Example:
  ```
  #mkdir /apg
  #mount /dev/dsk/c1t2d0        /apg
  ```

- Manually mount a file system readonly

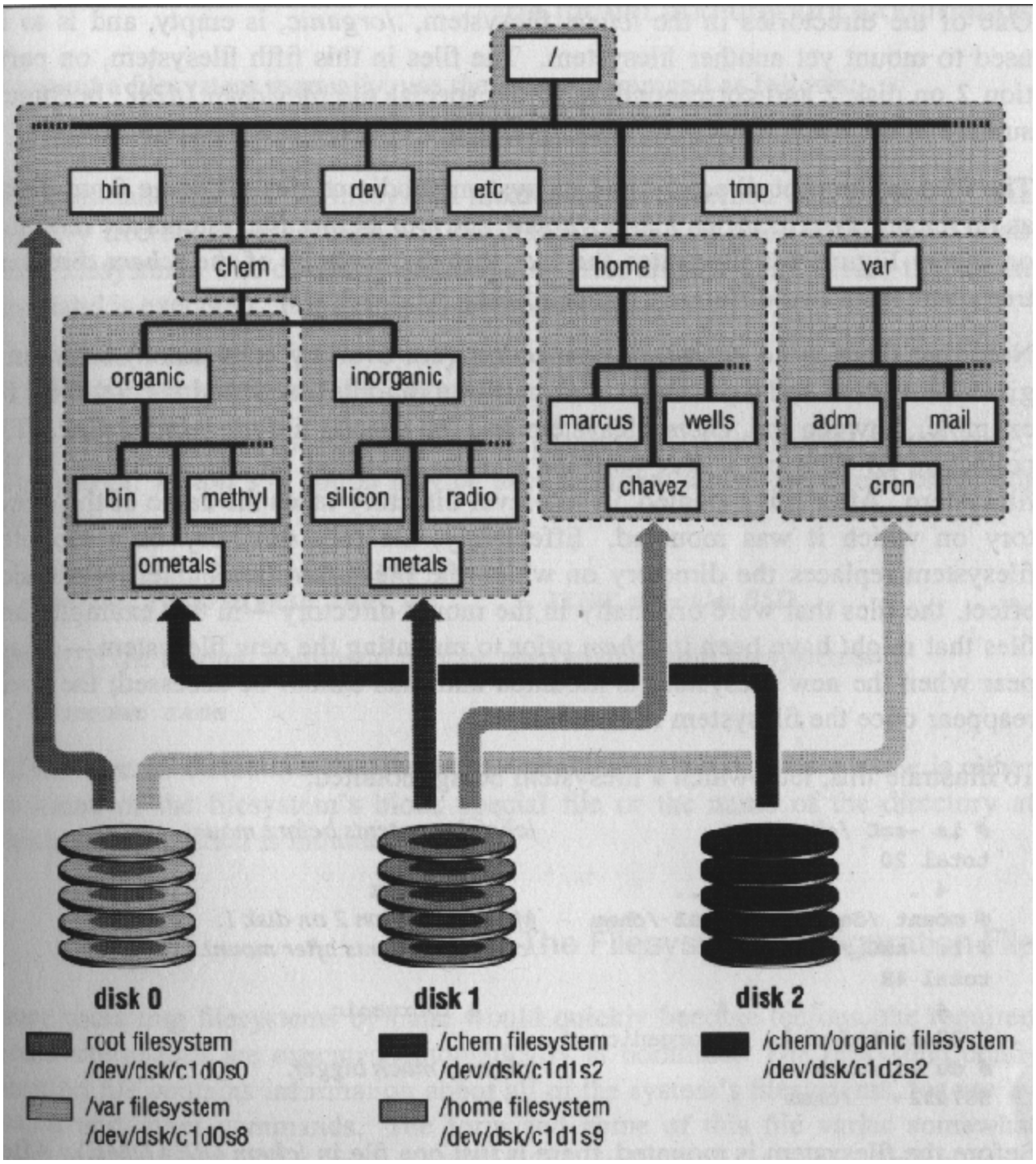
#mount –r /dev/dsk/c1d1s7   /mnt

- List all currently mounted filesystem
  #mount

- Umount a file system
  #umount name
  file system must be inactive

- Determine who is using a filesystem: fuser
  Identify user, process id
  Options: -u, -k

Example: how several file systems on three disc drives might be mounted.



File system configuration file

Mount command are executed automatically at boottime

/etc/fstab: SunOS, Tru64 Unix, HP-UX and Linux

format:

block-special-file mount-loc type opts dump-freq pass-number

type: vxfs, advfs, ext2,ext3, nfs, ufs, swap, …
opts:
rw
ro
suid
nosuid
noauto

more options for nfs

/etc/vfstab: Solaris
format:
block_spfile char_spfile mount-loc type fsck-pass automount? Opts

type: usf, s5, nfs, swap
fsck-pass: a number indicating the order in which fsck should check the filesystems.
Opts: rw, ro, rq, suid, nosuid, quota

Automatic filesystem mounting
–a option
Mount and umount only require either the mount point or special file name as argument
#mount /chem.
#umount /dev/disk1d

Using fsck to Validate a filesystem

- The fsck utility checks the filesystem's consistency, reports any problems it finds and optionally repair them
  Comparing the block free list against the disk addresses stored in the inodes
  Comparing the inode free list against inodes in directory entryies

- The five most common types of damage are:
  Unreferenced inodes
  Inexplicably large link counts
  Unused data blocks not recorded in the block maps
  Data blocks listed as free that are also used in a file
  Incorrect summary information in the superblock

- Fsck's scope is limited to repairing the structure and its component data structure.

- Disks are normally checked at boot time with fsck –p, which examines all local file system listed in /etc/fstab

Under BSD, fsck is run automatically

Under System V, fsck is run at boottime on filesystems only if they were not dismounted cleanly

If journaling is enabled, fsck simply rolls up the log to the last consistent state.

- Manually run fsck if there is serious error that fsck needs human intervention
  - You may be bring into single user mode
  - Check root partition first.
  - Rerun fsck until the filesystem comes up clean.
  - Do a good backup using dd to copy the whole disk

Disks

The geometry of disk

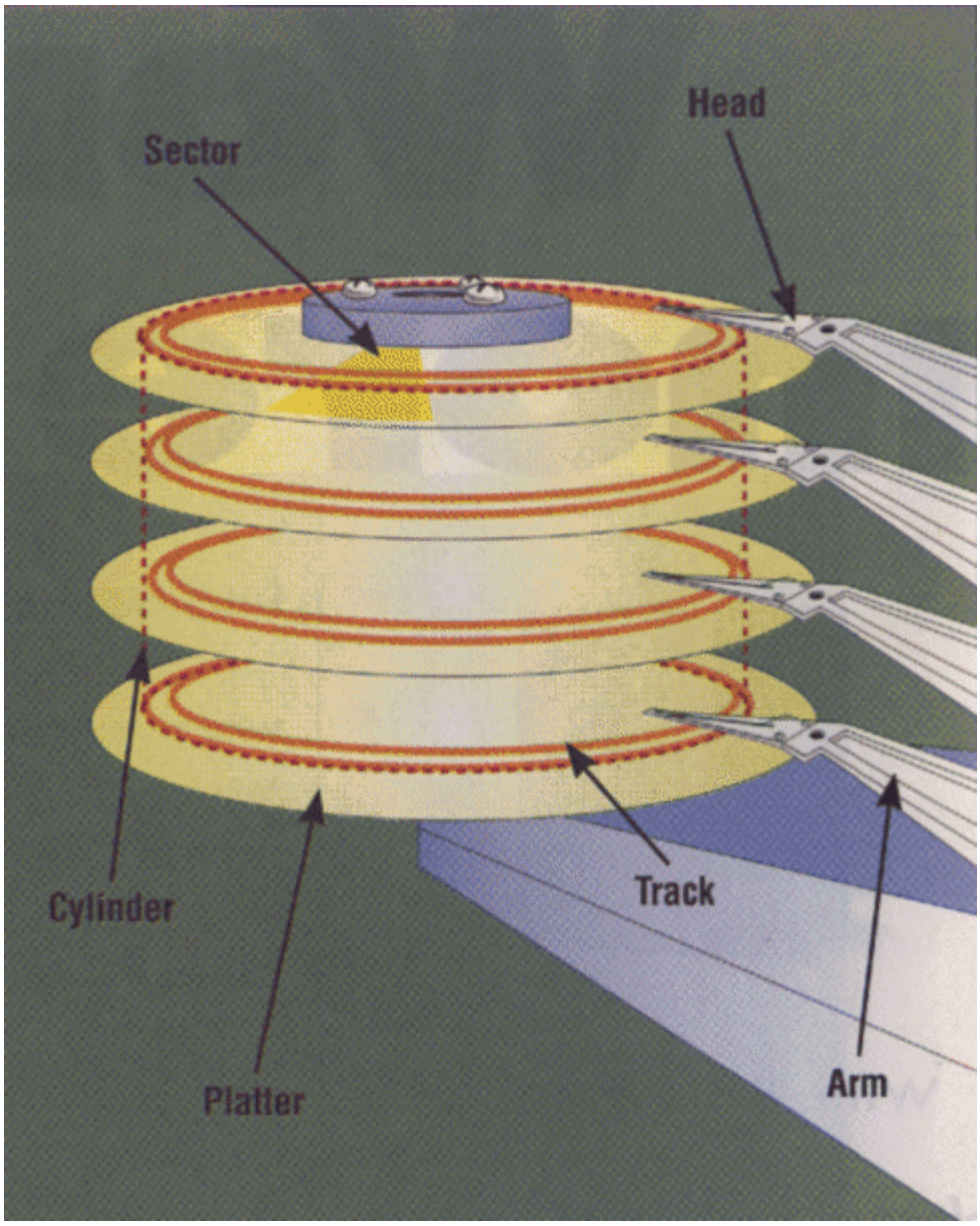
- A stack of platters coated with a magnetic film
- Data can be writing on both side of the platters
- Head moving back and forth, floating very close to the surface of the platters
- Cylinder
- Sector
- Track

Rotational speed and latency

- The faster platters spin, the less latency
- RPM from 3600 to 15,000 and more

Head movement speed and diameter of disks

- 14 inches 10 years ago, 5 ¼ inches 10 year ago, to 3 ½ inches and smaller …
- much slower that disk spins



Disk interfaces

Standard

SCSI (The Small Computer Systems Interface) – most command and widely supported disk interfaces on servers

IDE (Integrated Drive Electronics) – simple, low-cost interface for PCs.

Fibre Channel – high bandwidth and large number of devices that can be attached to it at once

USB (The Universal Serial Bus) – keyboard, mouse, removable hard disk and CD-ROM

SCSI

Available on CPU or peripheral board
Is used for disks, tape drivers, printers, scanners

Name	Specification	# of Devices	Bus Width	Bus Speed	MBps
Asynchronous SCSI	SCSI-1	8	8 bits	5 MHz	4 MBps
Synchronous SCSI	SCSI-1	8	8 bits	5 MHz	5 MBps
Wide SCSI	SCSI-2	16	16 bits	5 MHz	10 MBps
Fast SCSI	SCSI-2	8	8 bits	10 MHz	10 MBps
Fast/Wide SCSI	SCSI-2	16	16 bits	10 MHz	20 MBps
Ultra SCSI	SCSI-3 SPI	8	8 bits	20 MHz	20 MBps
Ultra/Wide SCSI	SCSI-3 SPI	8	16 bits	20 MHz	40 MBps
Ultra2 SCSI	SCSI-3 SPI-2	8	8 bits	40 MHz	40 MBps
Ultra2/Wide SCSI	SCSI-3 SPI-2	16	16 bits	40 MHz	80 MBps
Ultra3 SCSI	SCSI-3 SPI-3	16	16 bits	40 MHz	160 MBps

Connectors

scsi connector types and pictures.htm

SCSI uses daisy chain configuration,
so most external devices have two SCSI ports.
Internal SCSI devices are attached to ribbon cable where connectors can be clamped onto the middle.

Each end of SCSI bus must be terminated
Terminator
Auto terminating

Address
Each device has a target number that distinguish it from other devices on the bus
Address number is from 0-7 or 0-15
Address is essentially arbitrary generally speaking
Set the SCSI ID if available on the external device
Logical unit number (LUN) subaddressing
Disk array connected to one SCSI controller

Things can go wrong, keep the following in mind
Check differential

Check OS discover the new device fine
Check terminator on both ends
Check the length of cable, including the internal part
Never forget that your SCSI controller uses one of the SCSI addresses.

IDE

The controller is build into the disk, reduce the interface costs and simplifies the firmware

Also called ATA

ATA-2

Adds Faster programmed I/P and Direct Memory Access modes

Extends the bus's plug and play features

Adds a feature called Logical Block Addressing (LBA)

Over come the problem that BIOS can only access the first 1024 cylinders of a disk

ATA-3

Additional reliability, more sophisticated power management, and self monitoring

ATA-4

Ultra-ATA

Extend the bus bandwidth

IDE disks are used internally

Short cable length

One bus only supports two IDE devices

IDE devices are accessed in a connected manner

Connector

40 pins

pin1 to pin1

keep in mind:

New IDE drives work on older cards and old IDE drives work on newer card,

The cable length is exceedingly short

Old BIOS that does not see past the first 500MB

Upgrade firmware,

Replace motherboard

Drivers

Which is better, SCSI or IDE?

SCSI beats IDE in every possible technical sense

For best possible performance

Server and multiple systems

Connecting many devices
Particular features of SCSI

IDE is cheap and works well for single-user workstation

An Overview of the disk installation procedure

The procedure of adding a disk involves:

- Connecting the disk to computer
- Creating device files
- Formatting the disk
- Labeling and partitioning the disk
- Establishing logical volumes
- Creating Unix filesystems within disk partition
- Setting up automatic mounting
- Setting up swapping on swap partitions.

Creating device files

- Devices are presented as *special files* in /dev
- Devices are either *block* or *character* special files
- Any peripheral is seen by Unix as a device
- Even *memory* is a device (/dev/kmem and /dev/mem)
- Devices are created with mknod just as directories are created with mkdir
- Devices have *major* and *minor* number
- The *major* number represents the *device driver*
- The *minor* number represents the *instance of a device* of the type specified by the major device number

```
crw-rw-rw-  1 root    wheel     5, 10 Jul 30  1992 ptypa
crw-rw-rw-  1 root    wheel     5, 11 Jul 30  1992 ptypb
crw-rw-rw-  1 root    wheel     5, 12 Jul 30  1992 ptypc
crw-rw-rw-  1 root    wheel     5, 13 Jul 30  1992 ptypd
crw-rw-rw-  1 root    wheel     5, 14 Jul 30  1992 ptype
crw-rw-rw-  1 root    wheel     5, 15 Jul 30  1992 ptype
crw-rw-rw-  1 root    wheel     5, 16 Jul 30  1992 ptyq0
```

- HP-UX 11.11 “insf –a” will create all devices
- Tru64 5.0 “hwmgr -scan scsi”
- SunOS 5.x creates devices at boot time; use boot -r

Formatting the disk

- Write address information and timing marks on the platters to delineate each sector.
- Identifies bad blocks
- All hard disks come preformatted

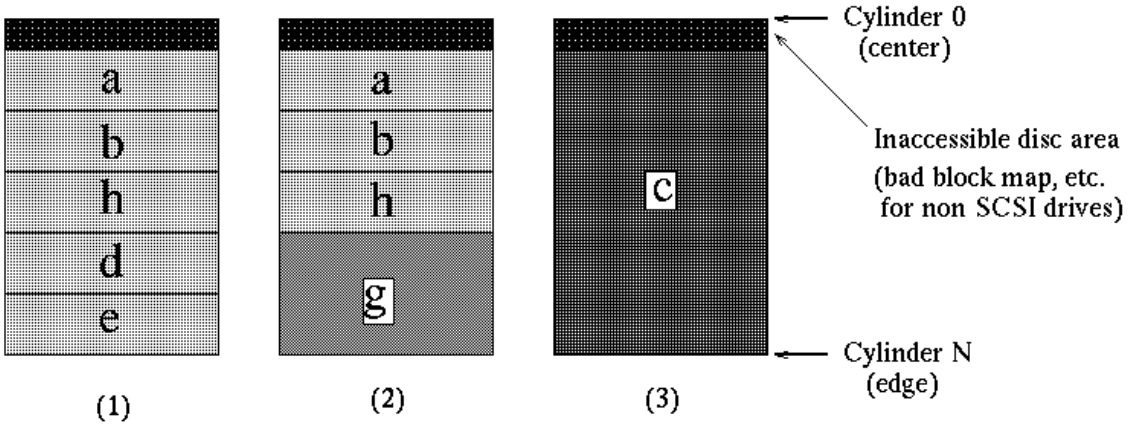
Labeling and partitioning the disk

- Disk must be divided into chunks called partitions or slices
- The partition table is kept on disk in a record called the label.
- The label usually occupies the first few blocks of the disk.
- Partitions can vary, but at least you will have:
    - Root partition
    - Swap partition
    - User partition
  - Some other considerations
    - Mirror the root filesystem on another disk
    - Add swap space if add memory to allow kernel crash dump
    - Split swap among several disks
    - Don't make partition bigger than the capacity of your backup device
    - Create a separate file system /tmp
    - Create a separate /var
  - Commands for partitioning/formatting drives
    - SunOS 4.x/5.x – format
    - OSF/1 - disklabel (gets partition information from /etc/disktab
    - Linux - fdisk
  - Commands for examining partition information
    - SunOS 4.x - dkinfo
    - SunOS 5.x – prtvtoc
    - OSF/1 - disklabel (gets partition information from /etc/disktab
    - Linux – fdisk

  - Example BSD partitioning
    - BSD systems usually use 8 partitions (labelled "a" through "h")

Example Eagle disc partitions

Partition	Cylinders	Typical use
a	0-15	/
b	16-86	Swap
c	0-841	Entire disc
d	391-407	Alternate Root
e	408-727	
f	728-841	
g	391-841	/usr
h	87-390	



Establishing logical volumes

- Supercharged version of disk partitioning
- Group multiple disks or partitions into logical volume that appears to the user as a single virtual disk
  - Concatenation
  - Striping
    - Higher bandwidth and lower latency
  - Software RAID5 (Redundant Array of Inexpensive Disks)
    - Striping plus checksum
  - Mirrored volume
    - Write both
    - Read split between two
    - Fail over and resynchronized

Creating UNIX filesystems

- After partitioning, you can create a filesystem
- Use newfs
- A BSD filesystem
  - consists of five structural components:
    - A set of inodes storage cells
    - A set of scattered “superblocks”
    - A map of the disk blocks in the file system
    - A block usage summary
    - A set of data blocks
  - Partition is divided into cylinder group, some structures are allocated among the cylinder group to reduce the need to seek.
  - A superblock is a record that describes the characteristics of the file system.
    - Information about the length of a disk block
    - The size and location of the inode tables
    - The disk block map and usage information

        - The size of the sylinder groups
    - Some commands
        - use "**newfs –N** " to see where superblocks are located
        - the system call **sync** flushes the cached superblock regularly
    - Performance
        - Fragmentation when it is close to full
        - Big block size
        - Inode table

Extending UNIX filesystems

- Utilities to extend it on the fly
- Example:
    - Tru64 AdvFS Utilities
        - Add a volume to a filesystem
        - Balance the file distribution
    - HP-UX JFS utilities

Removing a UNIX filesystem

- Umount the filesystem

Checking file system space

- **The command df**
- **The command du is for file space usage.**
- **CASE Example: Sparse files used by Oracle on HP-UX.**

Adding a disk to HP-UX

HP Provides Logical Volume manager (LVM)
Check the disk is visible to hardware and the kernel
Software configuration

Identify physical volumes: **pvcreate**
Create volume groups: **vgcreate/vgextend/vgdisplay**
Create logical volumes: **lvcreate/lvdisplay**
Create file system: **newfs**

My cheat sheets:

HP-UX LVM.doc
Tru64 Unix LSM filesystem createextenddelete.doc