File System and Disk Administation

Disks: a random access block device.
can support a file system.
One file system per disk partition.

Unix file system – A tree

Each disk partition contains a piece of the tree

File system consists of:
superblock, inodes, indirect blocks, data blocks

Basic concepts:

superblock: overall information about the fs
data blocks: actual file (and directory) contents
inodes: used to find datablocks of files

Information cached in memory:

superblock
data blocks (LRU replacement)
inodes (LRU replacement)

Data blocks

Contain data only, i.e., only file contents, nothing else

Problem: if you fragment by disk sectors you do too much seeking
Solution: group sectors into blocks
Block size can be selected (usually larger on large disks)

Block size issue:
Large blocks provide more efficient access time
Large blocks waste more space

Block size solution:
Provide fragments
divide some blocks, often in 1/8ths
Stuff small files or left over pieces into a fragment

All blocks on a file system use the same size:
superblock is in a block
inodes are stuffed several (≈ 64) per block

Inodes

There is one inode for each file
Each inode has a unique number

Inode structure:

Mode, `-rwx-r-x--x`
number of links
file uid, gid
3 times: create, modify, inode modify
file size (bytes)
the number of data blocks in the file
15 slots for addresses of the data blocks used by the file

When in memory this structure is extended to include, the name of the device which the inode came from, the block-size on that device and other information.

Inodes cache in RAM: a hash table

inode 1: points to the file that is the root (base) directory for the disk

Inode: Data Block Addresses Handling

15 slots for addresses

12 direct blocks (12 addresses)
1 Indirect block,
1 Doubly indirect block,
1 Triply indirect block.

Direct blocks: with 1K block size, up to 12K.

Indirect block: the address is a data block, but
the data block does not contain file information
it contains addresses of other data blocks
Extends number of allowable blocks in a file by ≈ 256.
Actual by the number of inodes that fit in a data block

Doubly indirect block: the block contains addresses
of blocks that contain addresses
of actual data blocks.
Extends number of blocks by about 256^2.

Triply indirect block:
Extends number of blocks by about 256^3.
File size limit: 16GB, except on large disk with a larger
data block size.

Directories

A directory is a file with a special format
A single inode points to this "file".
In this file we find an array of directory entries

Directory entry:
A pair consisting of an inode number and a (file) name
The inode is the one defining the file.
The name of a file is in the directory, not in the file itself.

Directory entry details:
inode number, length of entry,
length of file name in entry, file name.

The two lengths may differ because of padding.

Links

There are two types of links, symbolic and hard

`ln newname oldfile` – a hard link

The file/directory now has two names, the old name (`oldfile`) and the new name (`newname`)

Both names refer to the file.
They do not depend on each other
A file is not removed until *all* references to it are removed.

`ln -s apointer oldfile` – a symbolic link

`apointer` is an alias, it refers to any file that is called `oldfile`

Notice: `mv oldfile newlocation`
`newname` is a hard link and will still refer to the old file
`cat newname` – will display the old file

`apointer` is a soft link and will still refer to anything named `oldfile`
`cat apointer` – will say, "file not found"

The soft link is a name
a hard link refers to the file itself

Special Files

There is an inode for each device file
Device files don't use block addresses
First two address slots are used for the major and minor numbers

Other special files (communications ports)

Sockets (Unix domain only, not network)

Pipes (Named pipes only)

Block Groups

File system contains several block groups.
Each group contains inodes, data blocks
and a duplicate of the superblock
Allows the inode for a file to be close to the data blocks.

Free inodes and free blocks are managed by bitmaps.

Each group contains:
superblock
1 bitmap block for blocks,
1 bitmap block for inodes,
blocks for the inode table
data blocks

For each group there is a descriptor with:
the block numbers of the bitmap blocks
the free blocks count

Group descriptors are located after the main superblock.

When running:
a bitmap cache is used
(one per mounted file system).

Superblock

There is one superblock (plus backups) per file system.

Superblock structure:

The number of inodes, the number of blocks
the number of free inodes and free blocks
The address of the first data block
The size of a block and the size of a fragment
The blocks fragments and inodes per group
The last mount time, the last write time
the mount count and max mount count
The signature for the file system
The state (clean, dirty)
16 flags that say what to do on errors
Time of last file system check, max time between checks
The name of the creating OS (linux, hurd, masix, freebsd, lites)
Revision level of file system
Default UID and GID for reserved blocks

The information in the superblock is used to find everything else
data blocks, inode areas, group areas

Disks, Partitions and File Systems

One file system per partition.
Each partition is attached to the tree (as a subtree).
One partition is designated the root of the tree.
A partition takes its name from the attachment point,
it can be reattached elsewhere

The system adiminstrator (`root`) controls where disks are attached to the tree.

Examining the File System

`df` – disk free
lists the mounted devices, the usage statistics, and the attachment points

```
/dev/sda2 1556116 125640 1350152 9% /u2
```

`mount` – list attachments
lists the mounted devices and the pseudo file systems with their attachment points and the mount parameters

```
/dev/sda2 on /u2 type ext2 (rw)
```

`du` – disk usage
list the amount of disk used by each file or subdirectory

`dumpe2fs` – dump the superblock
list the contents of the superblock and the information about each of the block groups.

`fdisk -l` – format (parition) a disk
this option lists the partition table

`/etc/fstab` – file system configuration file
Tells the system what to mount and where.

```
/dev/sda2 /u2 ext2 defaults 1  1
```

Building a Disk

DOS/Windows terminology:
low-level format: build the sectors on a disk
high-level format: build a file system on a disk

Unix terminology:
format: build the sectors on a disk
make a file system: build a file system on a disk

`fdformat`: format a floppy disk, all information on the disk will be lost
Even after a disk is formated, you still have to build a file system in order to use it.

`mkdosfs` build a DOS file system on a formated disk
DOS file systems are fairly minimal.

`mke2fs` build a Linux file system on a formated disk The inodes are setup, the superblock and backups, the "root" directory of the disk, the `lost+found` directory
Required parameter: partition (device) name
Optional parameters: block size, number of inodes,...

Engineering issue:
Lots of small files: more inodes
Lots of large files: unused inodes waste space

Setting Up a Linux Harddrive (steps)

1) Clean out any obsolete partitions (`fdisk`).

2) Establish new Linux partitions (`fdisk`).

3) Construct file systems (`mke2fs`).

4) Test your setup (`mount`/`umount`).

5) Modify your `fstab`.

6) Test the reboot.

Setting Up a Transient (floppy/cd/pen)

1) Construct a file system (`mke2fs`/`mkdosfs`).

2) Test your setup (`mount`/`umount`).

3) Modify your `fstab`.
If you want users other than root to be able to mount these devices these must be done in the fstab.

4) Test your setup as a user (`mount`/`umount`).
(If it is a user-mount setup.)

Disk Partitions

`fdisk`: Interactive, type `m` for options.

The disk partition structure is usually expressed in cylinders.

0–500: Linux native
501–600: DOS
601–700: Linux swap

Warning: some BIOS (such as Sun Sparc) require there be an extra (fake?) partition that covers the whole disk.

0–700: disk

The x86 BIOS: allows 4 primary partitions
Secondary (logical) partitions subdivision inside a primary partition and were invented to get around BIOS limitation of 4 partition. Recommendation: Do not use secondary partitions with Linux.

UEFI firmware removes many of the limitations of the traditional x86 BIOS.

Your job: Make partition size decisions.
Create the partitions.

Partitioning a Disk

1) Run `fdisk` in interactive mode.
A list of commands and descriptions may be displayed by typing `m` (the prompt tells you this).

2) Delete any obsolete partitions.
You must delete the included secondary partitions before deleting a primary partition.

3) Create Linux partitions.

4) Change partition types as necessary.
When a partition is created it has a default type of Linux native.
If you want the partition to have a different type, such as Linux swap you must change the partition type (sometimes called the partition or system ID).
The type is a hexidecimal code.
You can tell `fdisk` to display a list of types and their codes.

5) Write and quit.

Building a File System

`mke2fs` will build a Linux native files system in a partition.

The only required parameter is the partition in which you want to build the file system. The command will pick appropriate parameters values for your disk.

You may override the defaults if you have special needs including:
The number of inodes.
The block size.
The fragment size.
The number of copies of the superblock.

You can also have `mke2fs` do a scan of the disk for bad blocks before it creates the file system.

`mkdosfs` and other mkfs's are available for various other kinds of file systems. If you are going to do this, you should have set the partition type appropriately with `fdisk`. That is, building a dos file system in a Linux partition is not recommended.

Mount (details)

With no parameters, `mount`, lets a user exam where disks are attached to the file system tree.

Sample output lines:

```
/dev/sda2 on /usr type ext2 (rw)
```

Partition 2 of the first hard disk is attached to the tree as `/usr`, this partition contains an `ext2` file system, it can be read or written `(rw)`.

```
/dev/fd0 on /floppy type msdos (ro)
```

The floppy in drive A is attached to the tree as `/floppy`, this floppy contains an `msdos` file system, it can be read, but not written `(ro)`

```
134.139.248.18:/slackware64 on /ourware type nfs (rw)
```

On the machine named `jaguar` (machine number `134.139.248.18` there is a directory called `/slackware64`. This directory and all the files and subdirectories in it are available over the network (`nfs`) and have been attached to our files system as `/ourware`.

Mounting a Disk

`mount`: with two parmeters, designates a disk and where it is to be attached to the tree.

```
mount /dev/fd0 /home/john/floppy
```

The disk `fd0` is attached as `/home/john/floppy`

After mount the "attachment" is invisible,
it is part of the tree

1) `fd0` must be a block device
2) `fd0` must have a file system on it
3) `/home/john/floppy` must be a directory

After the mount, any contents of the directory are hidden

Principle: mount to empty directories.

mount has options for example to mount read only:
`mount -r /dev/fd0 /floppy`
The manual entry for `mount` lists the available options.

Unmounting a Disk

`umount` – detach a disk from the file system,

The disk should not be in use.
No user should be `cd`'d to it.
No files should be `open` on it.

When unmounting, either the attachment point or the device is used (but not both).

For example to undo the mount from the previous slide, you may use either:

```
umount /home/john/floppy
```

or

```
umount /dev/fd0
```

The Filesystem Table

Most mounts use `/etc/fstab`

This table contains the list of disks, attachment points.
and mount options, (plus dump and check options).

```
/dev/sdc1 /u4 ext2 defaults 1 1
```

The device `sdc1` will be attached to the tree as `/u4`
The disk should contain an `ext2` filesystem.
default options should be used (see man mount).
dump is 1, check order is 1

All permanent disks should be listed

Transients may be listed, but use `noauto` option.

```
/dev/fd0 /floppy msdos user,noauto 0 0
```

an ordinary user can say `mount /floppy`
the floppy is locked in until a `umount /floppy`
transient disks shouldn't be dumped or checked

`fstab` also contains other information such as:
swap partitions
pseudo file system (like `/proc`)

Using `mount` With `fstab`

`mount` and `umount` can use `fstab`

You can mount all the disks found in `fstab` except those that use the `noauto` option with the command:

```
mount -a
```

Similarly you can unmount all disk (except `noauto`:

```
umount -a
```

If `root` uses mount with a single parameter (either the attachment point or the device the command looks up the other parameter to use in `fstab`:

```
mount /u4
```

or

```
mount /dev/sdc1
```

will look up the entry in `fstab` and do the correct mount

A user can use mount with a single parameter to use `fstab`, but the parameter must be an attachment point and that attachment point must specify the `user` option.

Caution: if the user doesn't own the attachment point there are some issues with writting to upper level of the device.

fsck

`fsck` – scans and repairs the file system for many problems

Will occur "automatically" at boot (check = 1)
Skipped if a `shutdown` or `reboot` did an orderly close of the file system
(even then after a fixed number of skips, it will do a check)

May be run manually
`fsck /dev/sda1`
file system should be unmounted

Five passes.

1) Blocks and sizes: inodes checked for consistency
2) Paths: follows directory paths.
Builds bitmaps of inodes and data blocks
3) Connectivity: inodes in use that refer to files/directories that weren't found in the directory tree
4) Reference counts: compare the inode link count against the number of directory references to those nodes.
5) Freelist: all unallocated blocks checked to be they are marked unallocated in the data block bitmaps

Device Names

Disk names follow certain conventions.

Linux:

`/dev/sda` – the first hard drive
(all drives are mapped to the SCSI command set)
(DOS/Windows drive `C:`)

`/dev/sda1` – first partition on first hard drive
(DOS/Windows partition `C:`)

`/dev/sda2` – second partition on first hard drive

`/dev/sdb1` – first partition on second hard drive

`/dev/fd0` – the `A:` floppy drive
`/dev/fd1` – the `B:` floppy drive

`/dev/usb/...` – appears when a USB device has been plugged in

BSD:

`/dev/sd0` – the first SCSI disk
`/dev/sd0a` – a partition on first disk
`/dev/sd0g` – another partition on first disk

partition names and order given by partition table