Filesystems, Files and Inodes

Joseph Hallett

January 12, 2023

Whats this about?

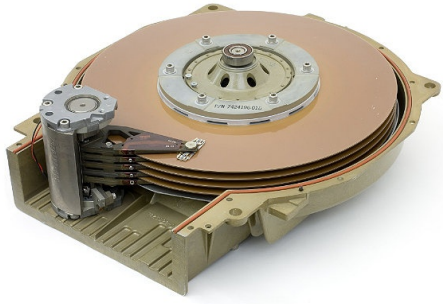
Are harddisks store all our files

- But what *really* is a file and how is the disk storing it?
- ...and how do we gain access to them?
  - (in a low-down way)

Basically we're going to give a quick tour of some low-level gubbins!

- Doing the topic properly could take an entire unit...
- ...maybe a whole degree?

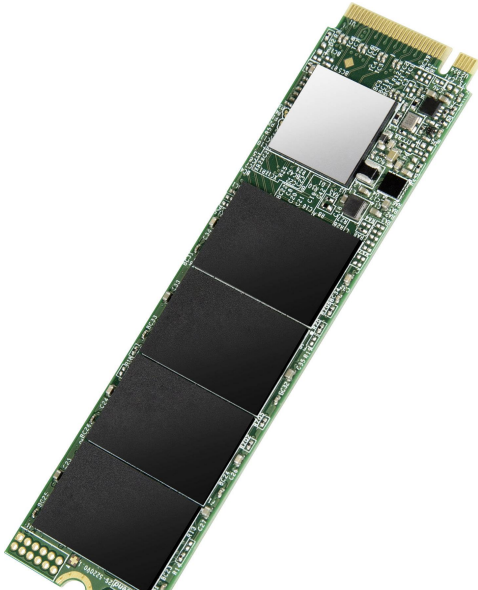
Harddisks?



Heres a harddisk!

- A *cylinder* of platters
- Each *platter* with a *head*
- Each *head* reading from various *cylinders*

Err... they don't look like that?



Heres another harddisk!

- Yes this is what they normally look like now.
- But we still pretend they have cylinders and heads.
- Yes you do occasionally still have to deal with cylinders despite them blatantly not existing.
- Isn't CS fun!?

Luckily for you...

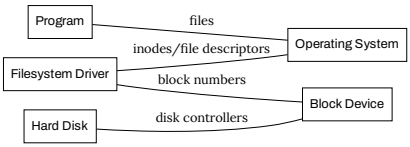
Unless you're writing filesystem drivers you *normally* won't have to deal with these details.

```
ls -l /dev/sd2c
```

brw-r—– 1 root operator 4, 34 Jan 12 08:53 /dev/sd2c
The operating system provides block files

- You can read and write into them like any other file...
- But you *probably* don't want to

Filesystems



As per everything in computer science its a hierarchy

- Different bits talking to different bits
- More and more abstraction

Paths

Filesystems start from a /

- The *filesystem root*
- Subsequent directories are separated by further /
- Just like URLs...
  [`https://www.bristol.ac.uk/engineering/departments/computerscience/index.html`]
  From the root of the webserver at `bris.ac.uk`
  - Go into the folder `engineering`
  - Then the folder `departments`
  - Then the folder `computerscience`
  - Then the thing called `index.html`

Special directories

`.` the current directory

`..` the parent directory

```
tree ../

../
|-- Git-1
|   |-- bristol.png
|   |-- dvcs.pdf
|   |-- git-stage-commit.pdf
|   |-- gitk.png
|   |-- linus.jpg
|   |-- linux.png
|   |-- slides.org
|   |-- slides.pdf
|   |-- slides.tex
|   |-- uk
|   |   `-- ac
|   |       `-- bristol
|   |           `-- cs
|   |               `-- SoftwareTools
|   |                   |-- Hello.class
|   |                   |-- Hello.java
|   |                   |-- Hello2.class
|   |                   `-- Hello2.java
|   |-- vcs.pdf
|   `-- vcs.png
|-- Git-2
|   |-- alices.pdf
|   |-- alicetree.pdf
|   |-- alicetreeanacbase.pdf
```

Interacting with files

High level C API (see `man 3 intro`)

```
#include <stdio.h>


FILE *fopen(const char *pathname, const char *mode);
size_t fread(void *buf, size_t size,
             size_t nmemb, FILE *stream);
size_t fwrite(const void *buf, size_t size,
              size_t nmemb, FILE *stream);
int fclose(FILE *stream);
```

(and all languages provide their own variants)

Low level POSIX API (see `man 2 intro`)

```
#include <fcntl.h>
#include <unistd.h>

int open(const char *path, int flags, ...);
ssize_t read(int d; void *buf; size_t nbytes);

ssize_t write(int d; const void *buf; size_t nbytes);

int close(int d);
```

(and all OSs provide their own variants of these system calls)

So what are these files really?

- The *filesystem* organises files into *paths*...
- The OS lets you interact with files via *file descriptors*
- Your *programming language* gives you a nice API for dealing with them

What actually is a file?

```
ls -i ./
```

```
22852856 bristol.png 22852858 disk.jpg 22852862 fs.pdf 22852863 fslayout.pdf 22852860
slides.org 22852864 slides.pdf 22852861 slides.tex 22852859 ssd.jpg
```

inodes.h

```
struct inode {
  LIST_ENTRY(inode) i_hash;          /* Hash chain */
  structvnode  *i_vnode;             /* Vnode associated with this inode. */
  structufsmount *i_ump;
  u_int32_t i_flag;                  /* flags, see below */
  dev_t  i_dev;                      /* Device associated with the inode. */
  ufsino_t  i_number;                /* The identity of the inode. */
  int       i_effnlink;              /* i_nlink when I/O completes */

  structfs *fs;                      /* FFS */

  struct cluster_info i_ci;
  struct dquot *i_dquot[MAXQUOTAS]; /* Dquot structures. */
  u_quad_t i_modrev;                 /* Revision level for NFS lease. */
  struct lockf_state *i_lockf;       /* Byte-level lock state. */
  struct   rrwlock i_lock;           /* Inode lock */

  /* Side effects; used during directory lookup. */
  int32_t  i_count;                  /* Size of free slot in directory. */
  doff_t  i_endoff;                  /* End of useful stuff in directory. */
  doff_t  i_diroff;                  /* Offset in dir, where we found last entry. */
  doff_t  i_offset;                  /* Offset of free space in directory. */
  ufsino_t  i_ino;                   /* Inode number of found directory. */
  u_int32_t i_reclen;                /* Size of found directory entry. */

  /* The on-disk dinode itself. */
  struct ufs2_dinode     *ffs2_din;

  struct inode_vtbl *i_vtbl;
};
```

```
struct ufs2_dinode {
  u_int16_t di_mode;        /*   0: IFMT, permissions; see below. */
  int16_t  di_nlink;        /*   2: File link count. */
  u_int32_t di_uid;         /*   4: File owner. */
  u_int32_t di_gid;         /*   8: File group. */
  u_int32_t di_blksize;     /*  12: Inode blocksize. */
  u_int64_t di_size;        /*  16: File byte count. */
  u_int64_t di_blocks;      /*  24: Bytes actually held. */
  int64_t  di_atime;        /*  32: Last access time. */
  int64_t  di_mtime;        /*  40: Last modified time. */
  int64_t  di_ctime;        /*  48: Last inode change time. */
  int64_t  di_birthtime;    /*  56: Inode creation time. */
  int32_t  di_mtimensec;    /*  64: Last modified time. */
  int32_t  di_atimensec;    /*  68: Last access time. */
  int32_t  di_ctimensec;    /*  72: Last inode change time. */
  int32_t  di_birthnsec;    /*  76: Inode creation time. */
  int32_t  di_gen;          /*  80: Generation number. */
  u_int32_t di_kernflags;   /*  84: Kernel flags. */
  u_int32_t di_flags;       /*  88: Status flags (chflags). */
  int32_t  di_extsize;      /*  92: External attributes block. */
  int64_t  di_extb[NXADDR]; /*  96: External attributes block. */
  int64_t  di_db[NDADDR];   /* 112: Direct disk blocks. */
  int64_t  di_ib[NIADDR];    /* 208: Indirect disk blocks. */
  int64_t  di_spare[3];     /* 232: Reserved; currently unused */
};
```

Simplified version of `/usr/include/ufs/ufs/{,d}inode.h` from OpenBSD

- Other implementations are similar
- Note that the `inode` doesn't have it's name in it...

Directories?

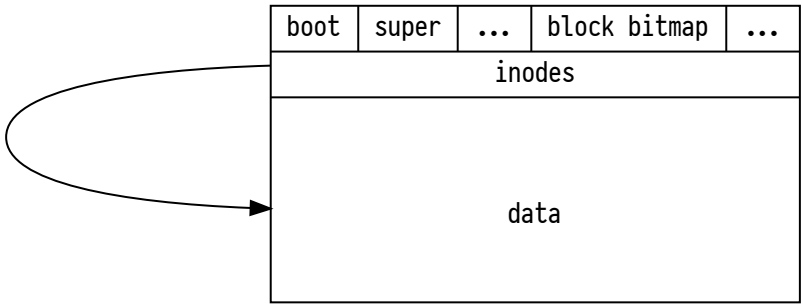
Just *inodes* with a list of *inodes* attached...

```
struct direct {
  u_int32_t d_ino;                /* inode number of entry */
  u_int16_t d_reclen;             /* length of this record */
  u_int8_t  d_type;               /* file type, see below */
  u_int8_t  d_namlen;             /* length of string in d_name */
  char      d_name[MAXNAMLEN + 1]; /* name with length ⩽ MAXNAMLEN */
};
```

And for those of us with small screens?

- Just a pointer to a region of memory on disk
- Each file has a unique ID *per filesystem*
- Permissions
- Metadata
- Link count (when 0 safe to reuse)
-

Disk Layout



Links

Two kinds!

Hard Create a file with the *same inode* as another file

- Only works on one filesystem
- Only works for files

Symbolic Create a file that contains as its data an alternative path…

- Works for everything and across filesystems
- Hope tools follow it rather that resolving the actual file
- (Almost all do automatically)

See `man 1 ln`

`ln file link` creates a hard link

`ln -s file link` creates a symbolic link

Other useful stuff

`rm`	decrease the link count of an `inode` (if it reaches 0 it'll be deleted)
`rmdir`	delete a directory (only works if its empty)
`touch`	create or update an inodes file modification times
`readlink`	show where a symbolic link links to
`fdisk`	create a layout on a disk
`fsck`	check the filesystem on a disk (useful after a crash)
`dd`	copy raw data to and from a disk

Useful filesystems to dimly recall

FAT	what USB drives use
exFAT	what bigger USB drives use
ext2	the old Linux filesystem
ext4	the more recent Linux filesystem

Summary

Computers have disks

- Theres an underlying data structure called an *inode*
- The OS will hide all these details away from you
  (*But you probably should know they exist*)