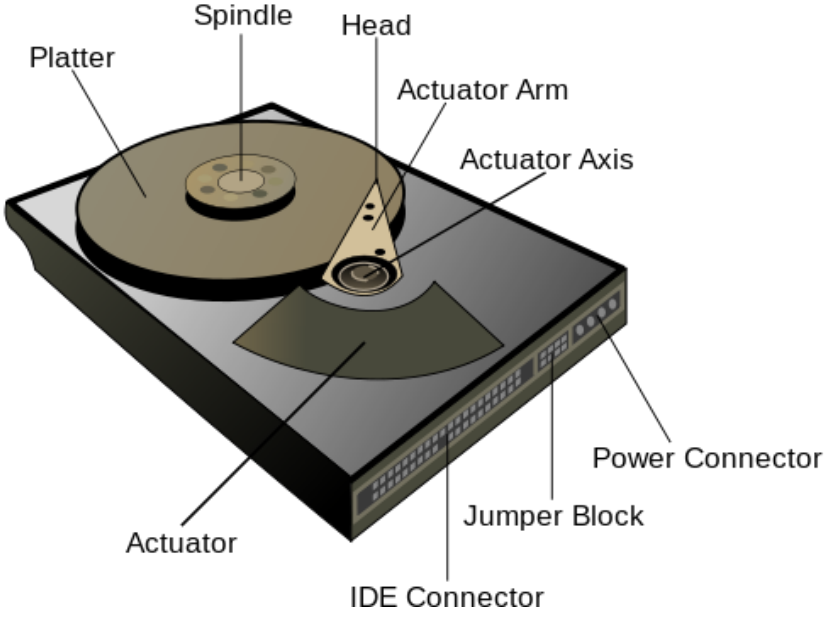
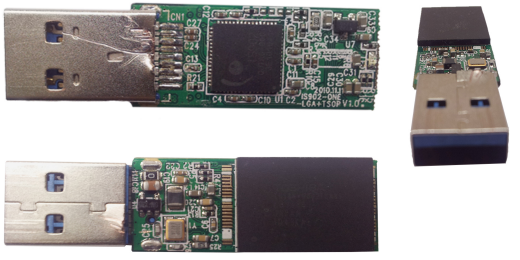
An Intro To File Systems

Pierre Rioux

ACElab Developer Meeting,
April 2015, Common Era

A disk device





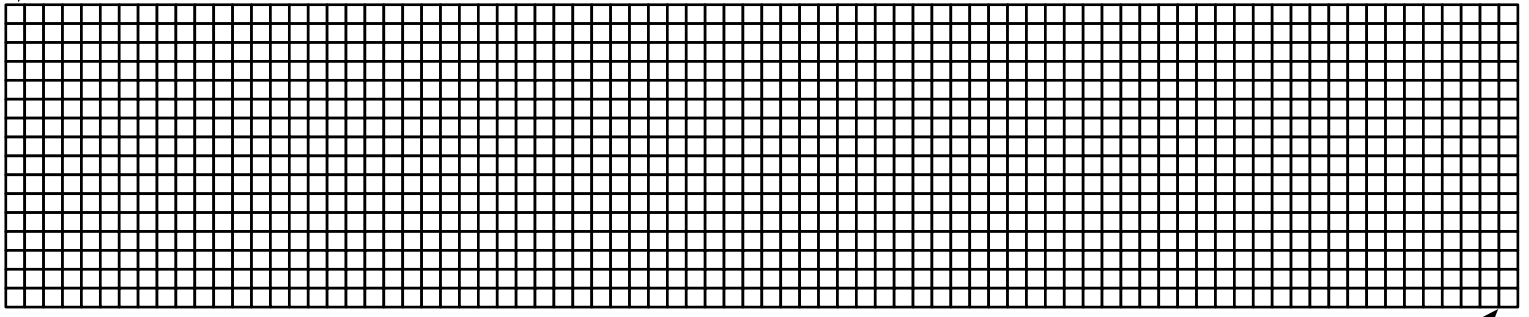
A block device

Basic functionality:

- The device stores a set of *blocks* or *sectors.*
- Blocks are numbered from 0 to n.
- Blocks contain a fixed set of bytes (old: 512, new: 4096).
- Reading and writing is made in units of full blocks.
- The device does not understand anything about files, directories, permissions, etc.
- Once connected, the hardware interface only shuffles blocks back and forth.

All Your Blocks Are Belong To Us

block #0



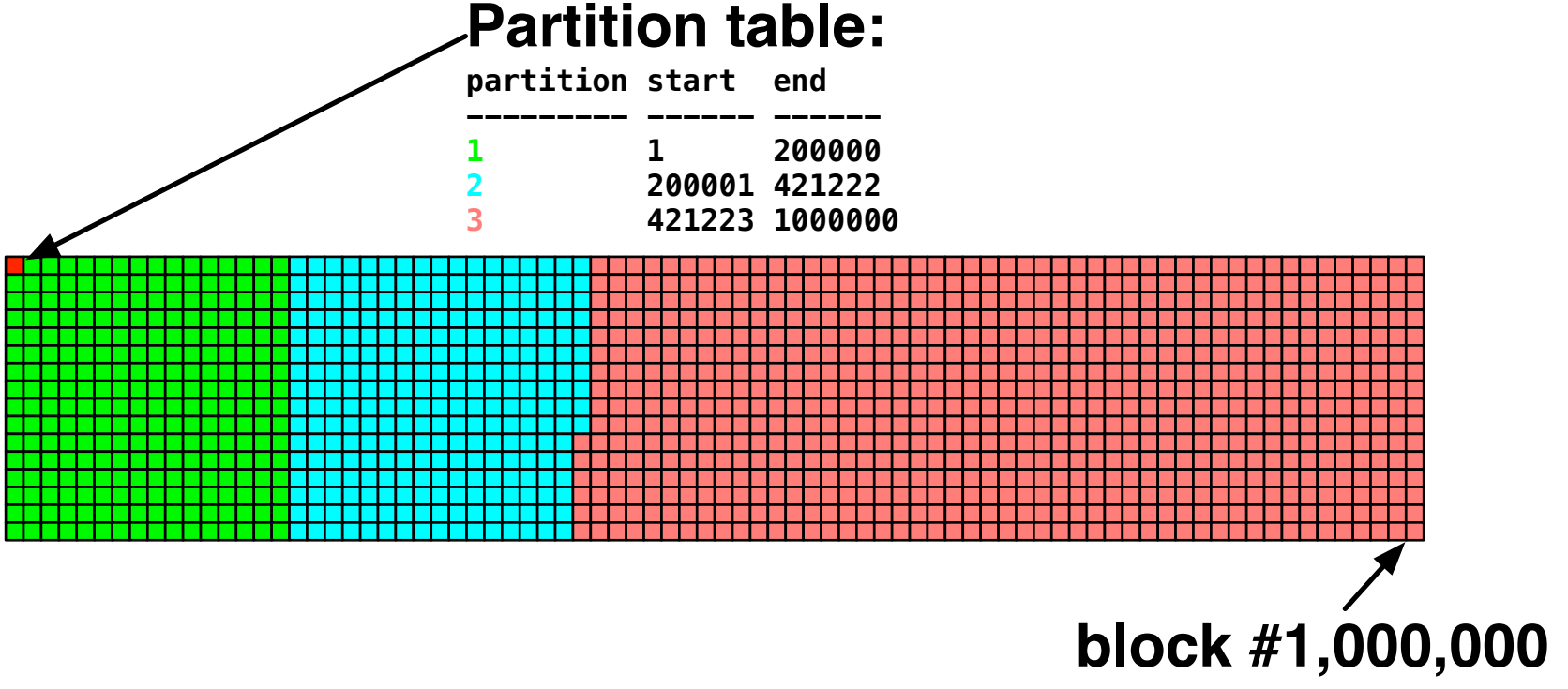
block #1,000,000

Each block contains 4096 bytes of data.

```
 H__PAGEZERO(__TEXTPP__text__TEXT3__stubs__TEXTDD__stub_helper__TEXTFF__const__TEXTII__cstring__TEXT0K60K__unwind_info__TEXTfOfOx__DATAPP__got__DATAP(PK__nl_symbol_ptr__DATA(P(PP__la_symbol_ptr__DATA8
PX8PR__data__DATAR(R__const__DATAR0R__bss__DATAT__common__DATAUH__LINKEDIT`0`'"0``h`He fSlnPQk /usr/lib/dyldMl.6I$*(0/usr/lib/libutil.dylib8/usr/lib/libncurses.5.4.dylib8/usr/lib/libSystem.B.dylib&e8)
 f+ f@rpUHHhHh]P5UHHFhHwhH];5UHHO`LA0HV`L9B0}HI8H9J8}]HhHh]4UHHHQ`LB0H~`L9G0}HR8H9W8}]HhHhHH]4UHHO`LA HV`L9B }HI(H9J(}]HhHh]c4UHHHQ`LB H~`L9G }HR(H9W(}]HhHhHH]4UHHO`LA@HV`L9B@}HIHH9JH}]HhHh]3UHHHQ`LB@
H~`L9G@}HRHH9WH}]HhHhHH]3UHHG`HH`HV`H9J`}]HhHh]P3UHHHA`HP`H~`H9W`}]HhHhHH]3UHHO`LAPHV`L9BP}HIXH9JX}]HhHh]2UHHHQ`LBPH~`L9GP}HRXH9WX}]HhHhHH]2UHAWAVAUATSHHIHH0H571.21t`Y>PH=6{1Ht8tH1"HUht@01tEt>{A1H=6%1
HtH0=+1Au$@E11/DAH@6L0t;HGwHHcH@@Du@D@@a}@@n@~@DAH=5H5t5|0D!H=5H55/!AADH=[5H5[5X/AADDEDAAD>DAD?D|k7Y?DQA G>D>I?K?E?DK?>>D==D>D=H=.4H5.4+.D=A DH=3H53-Dto>~g>DVY>DD{><H>H=3H53-D!H=v3H5v3s-t#>=>=D==D=xD=
DsDaH=F==D;F=D)4=6=0=D2==1DADH7HcHIDH)H=)2q,H,uH=2N,HH=29,HH^+H=1LLE+H<H=1L/+H;H=1L+H=1H;L+H=1H;L+H=1H;L+H=1H;HuH=1H+H;H=;tHn;HtHt;E1=;tF;L-]"L+L+H=1#+H DB;DDB;DD<;D5S;EY;:DDAEEAADDDtBEu2H}H5:+H:HH?H7HHHw:Hj:uE
tH]Et!H~uHWu-HT|=!:u?=d:u?=:u?HEX=9uG=@:uG=9uGHHkHQH8H;uH5GHD,=U8(q//////////////////G//fy///////////E[m/7M,x4UHAWAVAUATSPA}1H6HEHD'HHAEH1k'1H=6uHH[A\A]A^A_]H'6uDEH6'IMALSAD
$XwwIcLA|$8<^H=,H5,&EB4:A|$8O~'IHH=,LH0'5fA|$VtA|$h.u5=5uGIt$0H=,0&RIhH=Y,L09'H=+H5+%}|It$0H=J,0&i5Hu%IH=+H5+%$tt$4u MLtfxXuH@H@HuLL4uMtHL%H%IM_$D0Hb%D0$8uH[A\A]A^A_]H5I+H5x+0$..UHH?fOX1ftJH6fVXft=ftf
u]2f9t(fVu!4uftfu]H3]UHAWAVAUATSHHIHH.HHEMp44u=4H=+]$HHHH0H8H HH@H(5;,H$H|$$H:u@f0:HH@HHSt5H{:u:u0SQAHAHIHuy:uf0HHHHH8H(H\$ H|$Ht$HT$H$H5d)H H@L0LH0#>3HHcHH H@H0HHH8H(=2u21H HtHHHuHH H@HtHHHuHH@1H0HtH
HHuHH0HHHtHHHuHHHLH1MHIEfAEXftfu2A}8"IuhH=g'H0*"IE0HufuU0uA}h.uA@tAuBH;8vH8H1>1tI}h4H;8vH8M}`IGhH;@vH@IGH; vH AGH;0vH0IG`H;H~HHHIGhH=0EO=0uD1!IA1U HKH]IL1H}&IM0ZEOH]H1M09L II9vDLHHb HH9v1=/tUAt>H}8uH:
H=% H[HH HcH9vM4J|0$tHH)H{H;HI|H{HAGG%=`u=.t J]3H{LLSLyI] MmMHaHPd8h`HHXL@L}L5$L1M0pLL$lp(txL L10LnL;$|L0L10L<ELHL1M0LEEHP,,=r~tH{ H[HuH_'HH;Eu.HH[A\A]A^A_]H5l#H5##0=(2<GRUHAVSIH=D"H5D"AtGIfxVt=H,H,t)
IFHz,1H1HHH="H0IH[Ht#H{tAVAv,H[A^]UHAWAVAUATSPIII^`E1H0,8tHSH="0GAH3,8t8HChH]H+HH9HHH=M"DH0AEH]EAL%+A<$tHExAIhLA<$t:Eu5HB+H8L5cLwH4+H8L`DH+8HEtxlH[A\A]A^A_]UHAWAVAUATSHIL0%HHEM,$fA}VtBH+H*t.ID$H*1H1H
HH=!H0M,$M,I}M}`H*8tIWAt$,H= 0H*8t.IGhHH#*H9HHAt$H= H0\AL@HM} MehfA}VMt6IELH(L$$HHL< 0IL8LHLHHPHMt'H1H`uHHPL11aHHHHH_HHH^IL0~^@HPu^ ^+L H@D@AL$0H(H(LttH=6L0StEL$(HCH$HEL$8tH$H=AD$(H{H|$D$H$H=L0HR(8L@t
H5At$ H=0iAG@%=`u AWr$H="AD$4A|$uIw`Hd2H=0")MG`H=H0H'8LuH'8uH~'8uI0I I@IPPH['8tA,H8DH3'8t>,t5H&H8HHH&H8HH1'8LHtAAG%f=H8fA}VuL1LFI%IELH(H$LL0LHHu8MMHd H\8HML5H0!HcH=0:H.H/&8HHHH)LHL LHH8HHM~wH~rH8HILL
11E1AUIHLQA4HLhLNt0I9rH8LHMHP/LHH$8H@fXHHP1Hf@DE1A3HH$LLL0HH|:HHp}HHhtHfIMIM>H#8uyLHHu_u<jHtDHH$LLLu<HLLLx|HBtuH:H6HpLH9H+HEH=DLI0LmE1AH AtHht2Edt+EH5HHEITH=0AIAuL~/Bt+Hpt1F$+t+tEH5zH|HEJT+H=g0VAHpuF1H
;HPH,AHPL0LHMmM(HHH;Eu%H[A\A]A^A_]H50UHSPHHHq!9t7H]HHA A<H=H0iH=0WH[]UHAWAVSHhL=IHEH}=y98dH= u1XHu H 8tH~H=`HE{H=H5HMHH) tH9%H9~u)H9~HH9}H&H)H*H.HEH}(LuLPHH}HH0LIH;EuHh[A^A_]UHSPw'w._w7tF uEyuNju?[`u&
B;@utBtA#1It'tt_H[]1UHHHt]HH8t]x]UHH8t1@ut'?wuK|Qw@u4/:tu=#@u%1@It*s]UHAWAVAUATSPIM&ME1L=I|$uImI|$huIcHI|$HAHpH9rE1AVAv,LfDI|$Md$tL0AM$$MxEtH[A\A]A^A_]H[A\A]A^A_]UHAWAVAUATSH(IHD9DAFt9~7IcvH="HEHHu10L
I1HMH@Ht!HxtHUHcHHHUAFH9tAN,DHtAVD]H:DA]A!C?H9}LH([A\A]A^A_]NAHEEIHRfxVt4Ht)IFH1H1HHH=?H0D}Lu}1AEH9DDDeAADeAAMcHJ<HEPp,&HDUAHUA9}gDU#UD9MHjDAu.H78 EJEAT#UD9~A9}DA9RE;EH([A\A]A^A_]H=H5H?UHAWAVAUATSH8HH
HEHEHHEE11HHHEHHIHDLlHHE9~HUHEHHME1HuIIBD=IfAw6AffEHHH=NybwALwAAE2<xuHH8H50AfDAAHuHHEHfAH8[A\A]A^A_]UHAVSAHHEHEHH8HHH8HD1%D[A^]UHAVSH@L5A|tHyH8HzAt+H=H81qHtHJH^ATt.HH81=HtHH[A^]&[A^]UHD1]UHH@}Hu1H]UHA
WAVAUATSHIHHHEW)E)E)E)E)p)`)PHALHHLLHPrHHHHtS~AMA<$ZIAuIHcLHwHD<D!{eW)E)E)E)E)p)`)PA?HI3HH=1L0HHHH;MuH[A\A]A^A_]UHAWAVAUATSHIHLHHEW)E)E)E)E)p)`)PE1<?2W)E)E)E)E)p)`)PAIHLLHPHHHHtHcLHwHD<!u?IA~A
MA}tIAuILAV?LAHHH;EuDH[A\A]A^A_]UHAWAVAUATSHIHHHEW)E)E)E)E)p)`)P1Lc1AMHLLLHPIMIt}It6HcLHwH?D<D
```

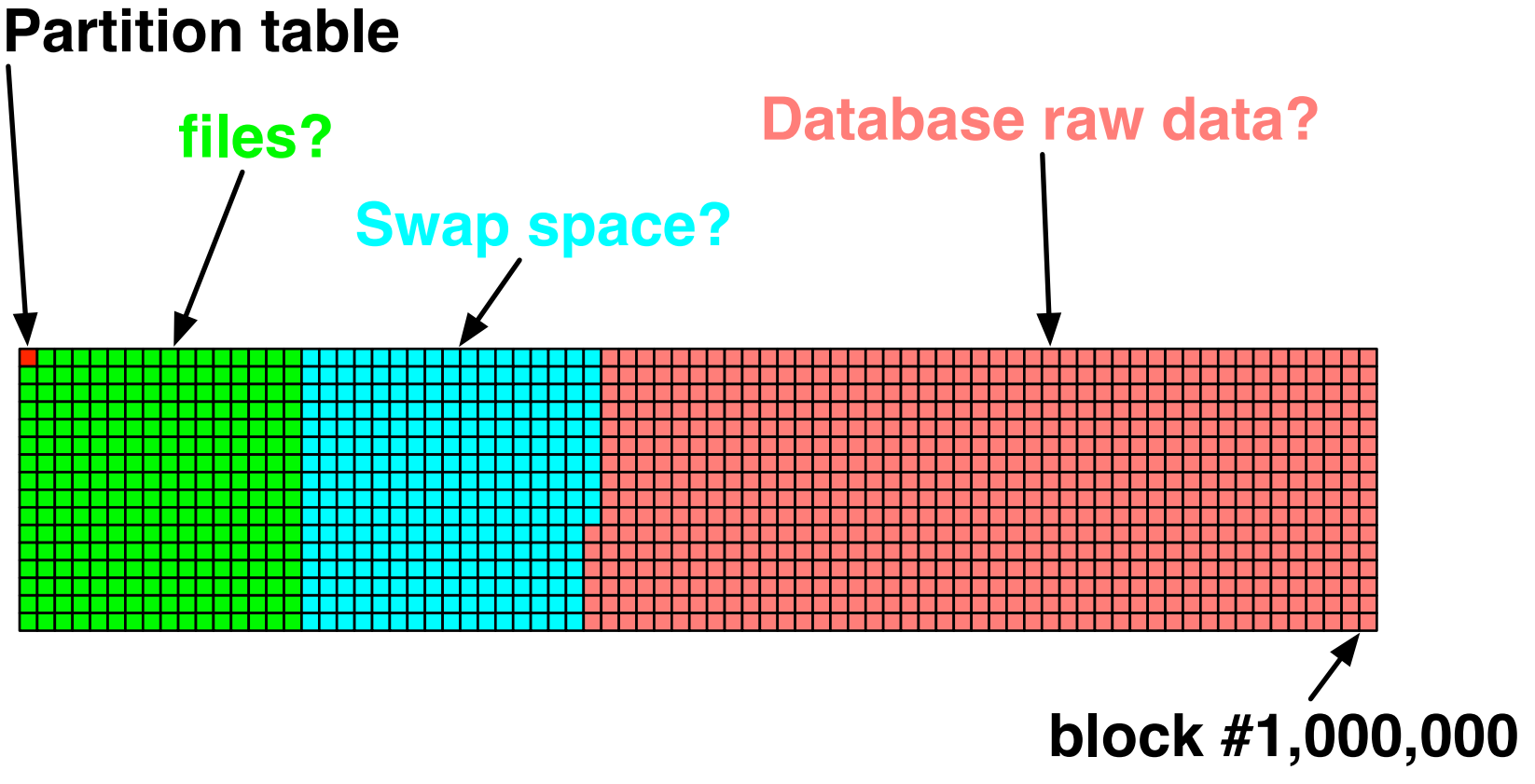
Partitions

- They are a *convention.*
- There are several ways to record them (dos, gpt, apple...).
- Often recorded within the first few blocks.
- The device does not understand them either.



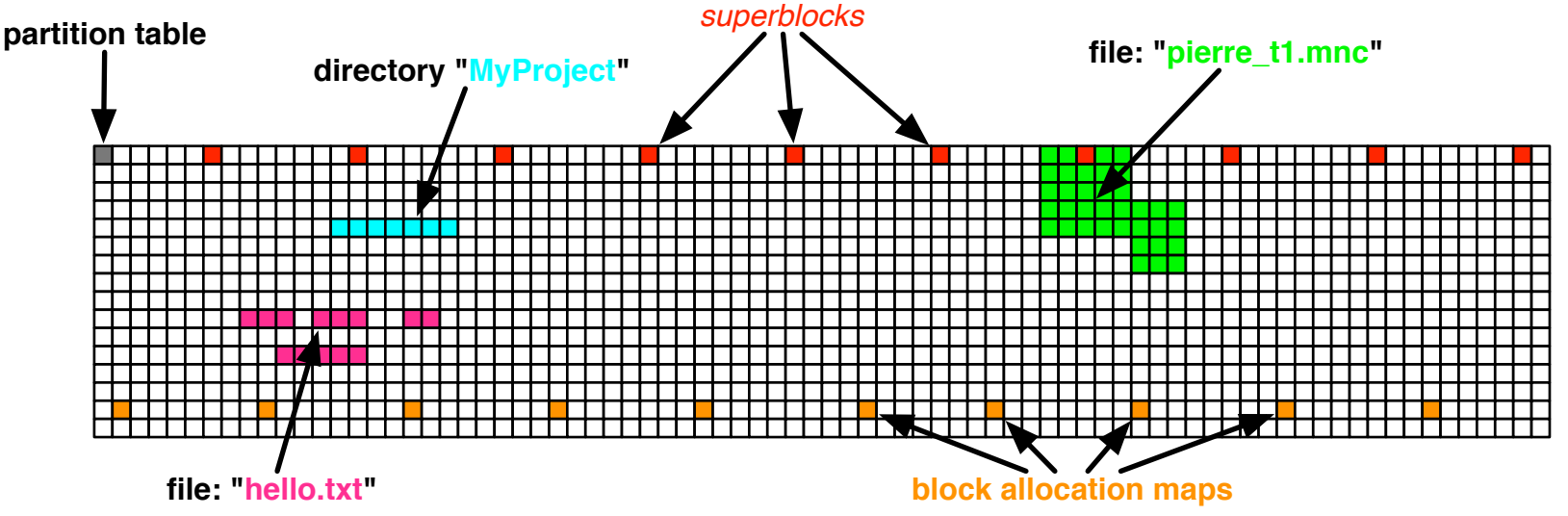
Partition roles

- Roles are not enforced by the device.



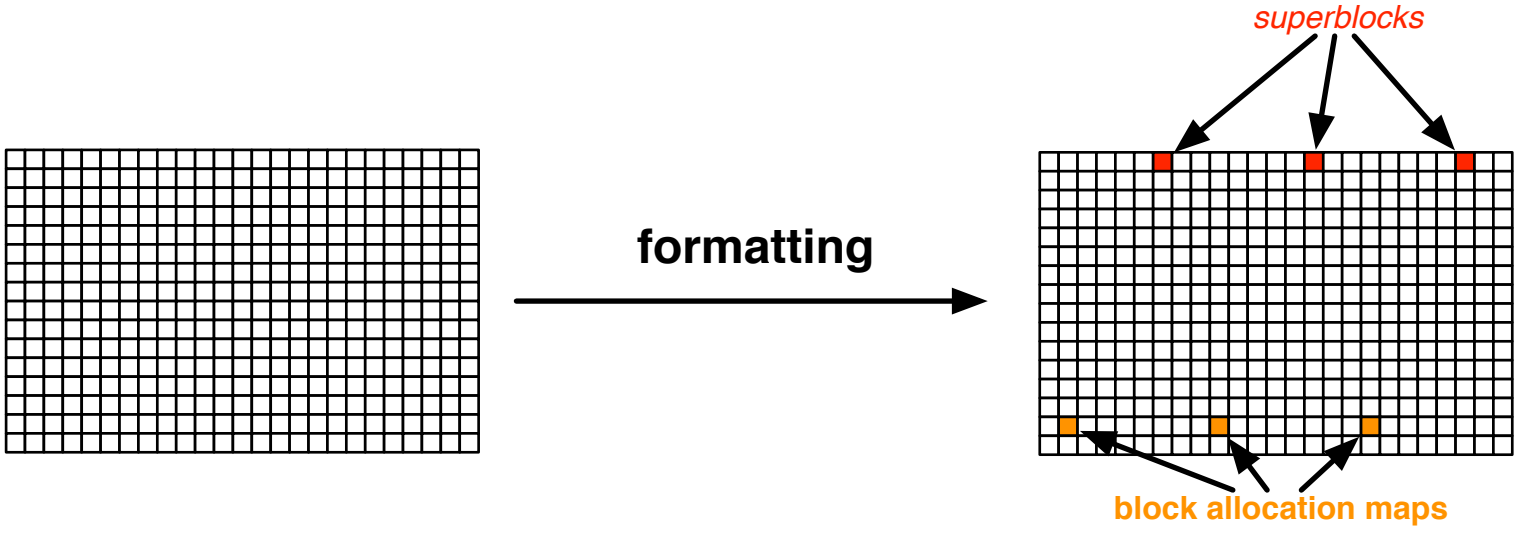
File systems

- They are *software components* of an operating system.
- They manage *files*, *directories, etc* within a partition.
- They keep their structures persistent by using special blocks within the partition (think about power failures!).
- By extension, we use the word *file system* to talk about the set of files and special blocks within a partition.



Formatting

- Given a partition with no particular data on it, installing an initial file system is called *formatting*.

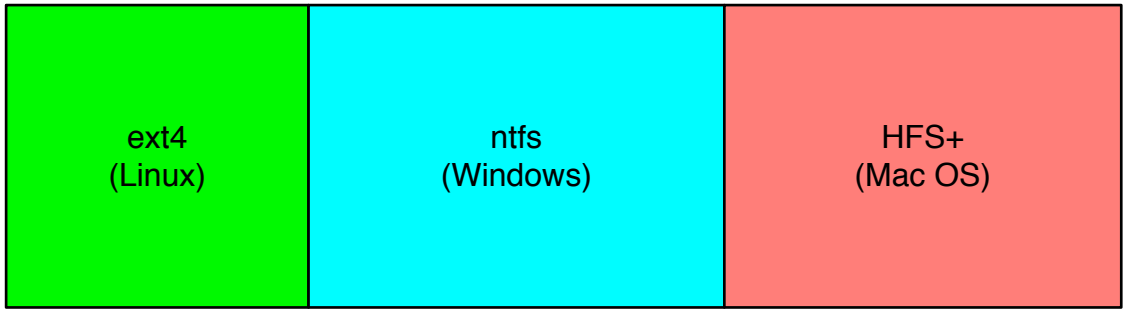


Types of file systems

Names	Years	Typical OS
fat, fat16, fat32	1977, 1984, 1996	Windows, USB keys
ext, ext2, ext3, ext4	1992, 1993, 1999, 2006	Linux
ntfs	1993, 2001	Windows
btrfs	2007	Linux, UNIX
ufs	1994	UNIX
hfs, hfs+	1985, 1998	Mac OS

Hybrid disks

- A single device can have multiple partitions, each with a different type of file system.
- The data in each file system can be read by the same computer, as long as it has the appropriate file system software component in its operating system.



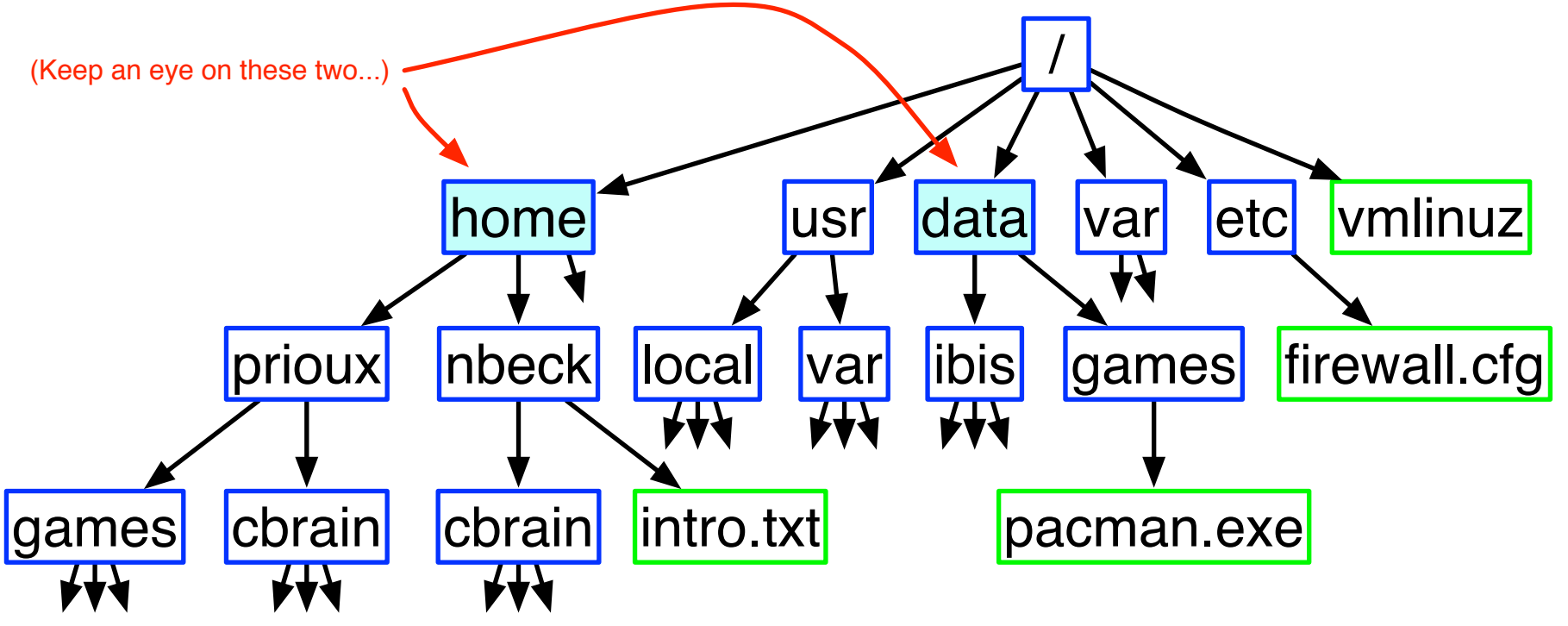
A weird USB key; 3 partitions and 3 file systems types

The UNIX file system 1

- UNIX-like operating systems present all files under a single tree rooted at "/".
- / itself is the main directory of the filesystem of the boot disk.
- The entire set of files visible under / is often called ***THE*** file system, even though it may be comprised of several file systems.
- It is organized in a tree structure.

The UNIX file system 2

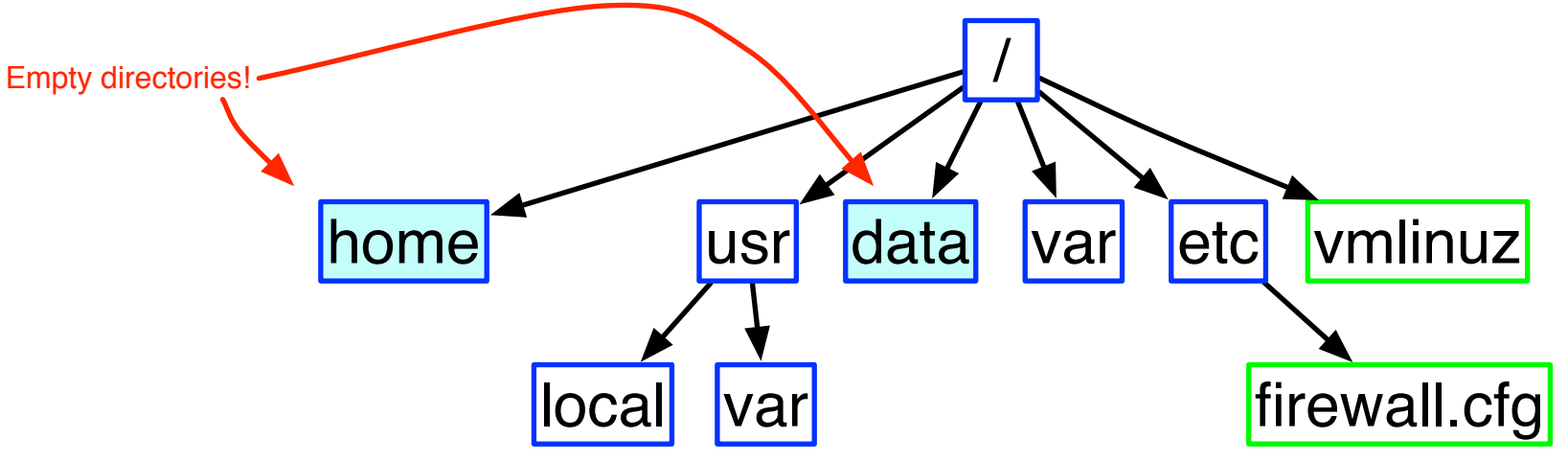
- ... we're all familiar with this view, right?



Directories are in blue, files are in green.

UNIX mountpoints 1

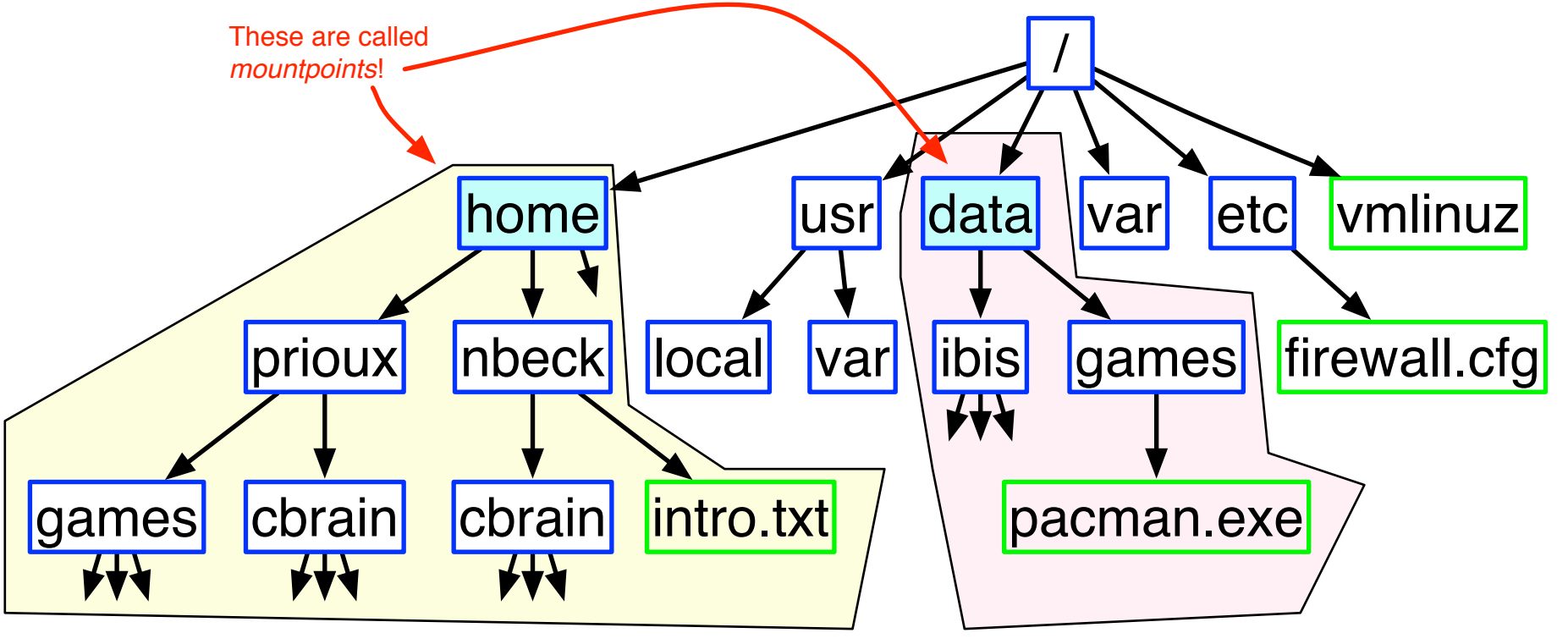
- When the system boots, the file system might be partial...



- These are the files and directories visible on the *root* file system.

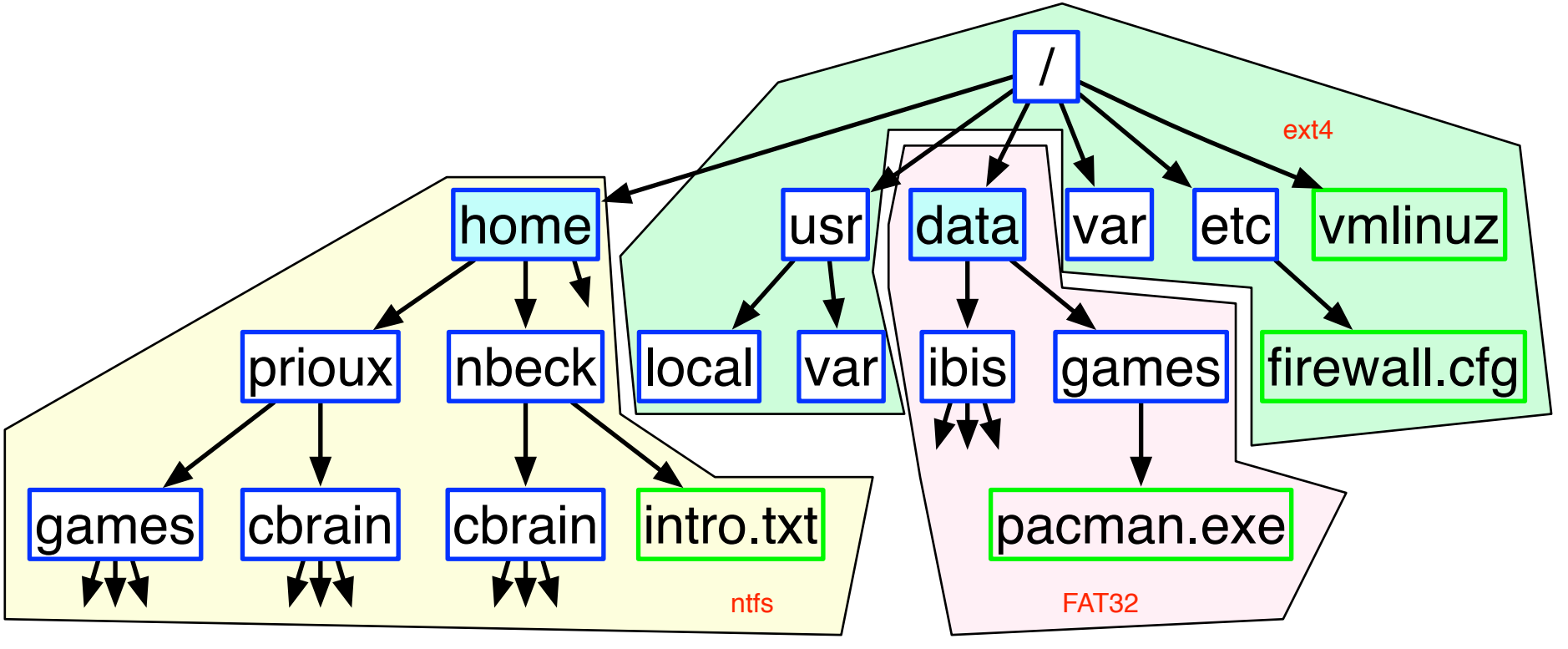
UNIX mountpoints 2

- Other file systems are attached to *mountpoints*.



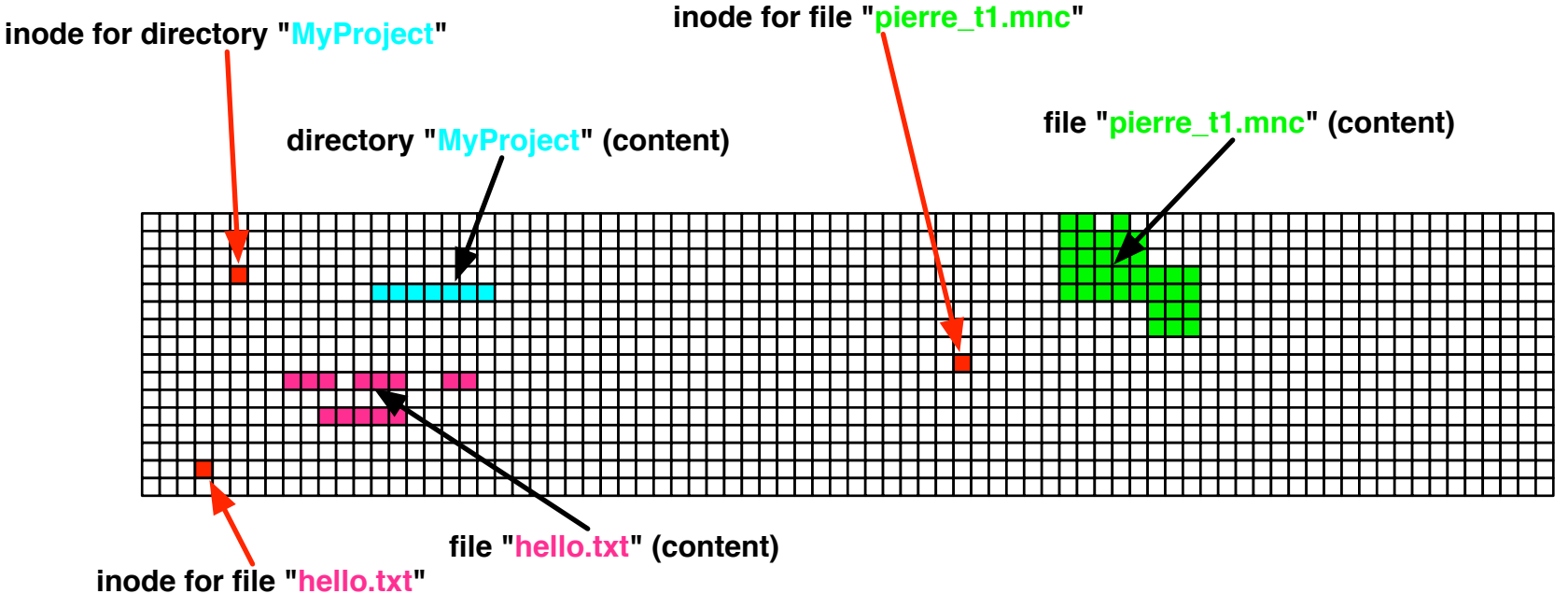
UNIX mountpoints 3

- File systems *mounted* can be of different types.



inodes (not an Apple product)

- Given a UNIX-like file system, each file or directory on it has an *inode*.



Content of an inode

- An inode contains many pieces of the information familiar to all programmers:
    - user ID
    - group ID
    - size
    - type
    - file permissions
    - several timestamps
    - list of blocks with contents
    - link count (more on that later)

Quick Quiz: something important seems to be missing!

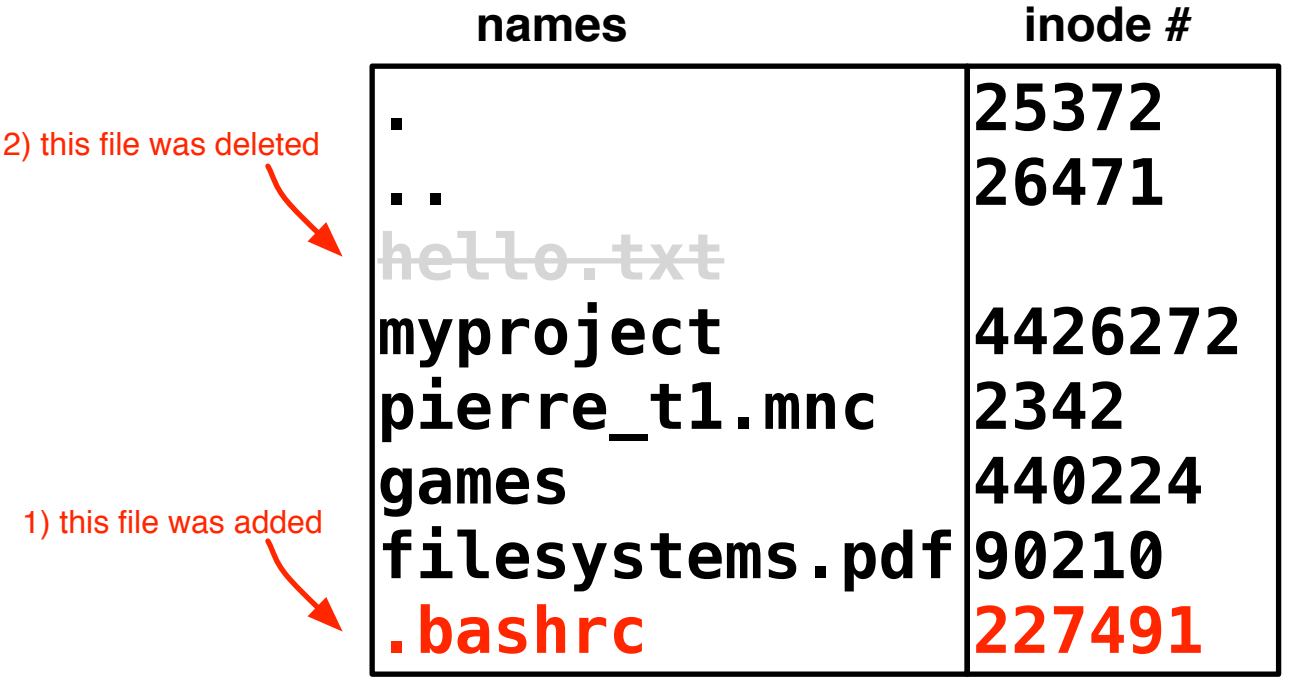
Directory content 1

- The bytes in the blocks storing the *content* of a directory are organized into a flat list of entries.
- Each entry is a simple pair: (name, inode number).
- They are not stored in any particular order.
- You can't tell which entry is which type!

names	inode #
.	25372
..	26471
hello.txt	11242
myproject	4426272
pierre_t1.mnc	2342
games	440224
filesystems.pdf	90210

Directory content 2

- New files are added at the bottom, making the list longer and longer.
- File that are deleted leave a hole in the table.
- There is often NO optimization to compact this list if there are too many holes.



Directory content 3

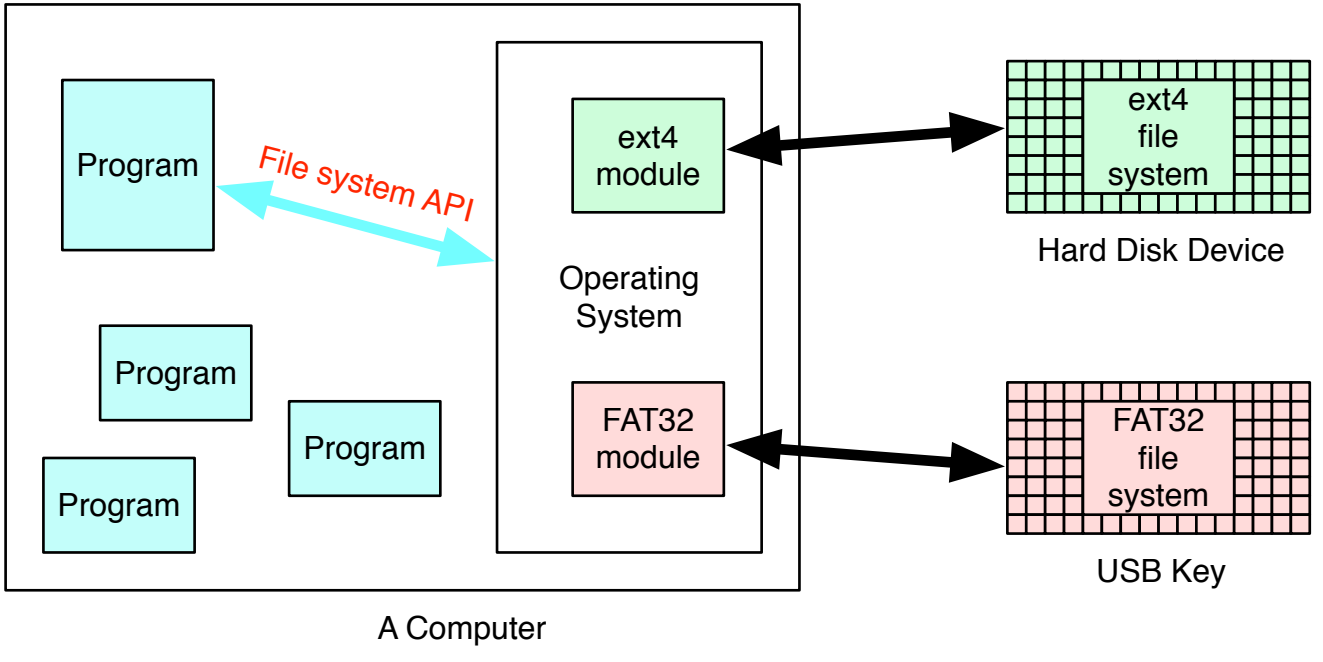
- If millions of entries are put in a single directory, the list becomes enormously inefficient.
- The operating system scans the list linearly from beginning to end whenever an operation is performed.
- "echo abc > def" must scan for file 'def' anywhere in the directory, if it exists, and otherwise create a new entry.

Directory content 4

```
root@brainstorm cbrain # ls -la
total 108
drwx------  5 cbrain cbrain 4096 Apr  8 12:29 .
drwxr-xr-x 10 root   root   4096 Aug 20  2013 ..
-rw-------  1 cbrain cbrain 3402 Apr 15 17:05 .bash_history
-rw-r--r--  1 cbrain cbrain   33 Apr 16  2012 .bash_logout
-rw-r--r--  1 cbrain cbrain  176 Apr 16  2012 .bash_profile
-rw-r--r--  1 cbrain cbrain  124 Apr 16  2012 .bashrc
-rw-------  1 cbrain cbrain   35 Nov  2  2012 .lesshst
drwxr-xr-x  5 cbrain cbrain 4096 Nov  6  2013 .mozilla
drwx------  2 cbrain cbrain 4096 Apr 17  2012 .ssh
-rw-r--r--  1 cbrain cbrain 5988 Apr 24  2014 SSLCerts-2014-04.tar.gz
drwxr-xr-x  2 cbrain cbrain 4096 Sep 22  2014 .storage_system
-rw-------  1 cbrain cbrain  956 Jul 25  2013 .viminfo
-rw-------  1 cbrain cbrain  292 Apr  8 12:29 .Xauthority
```

```
root@brainstorm cbrain # ls -1fi
28737551 SSLCerts-2014-04.tar.gz
28737555 .storage_system
2 ..
28737545 .ssh
28737547 .lesshst
28737549 .viminfo
28737597 .Xauthority
28737543 .bash_logout
28737537 .
28737538 .bash_profile
28737544 .bash_history
28737542 .bashrc
28737539 .mozilla
```

API


Program
File system API
ext4
module
Operating
System
Program
Program
Program
FAT32
module
A Computer
ext4
file
system
Hard Disk Device
FAT32
file
system
USB Key

API for files 1

- open()
- read()
- write()
- close()
- unlink()
- seek()

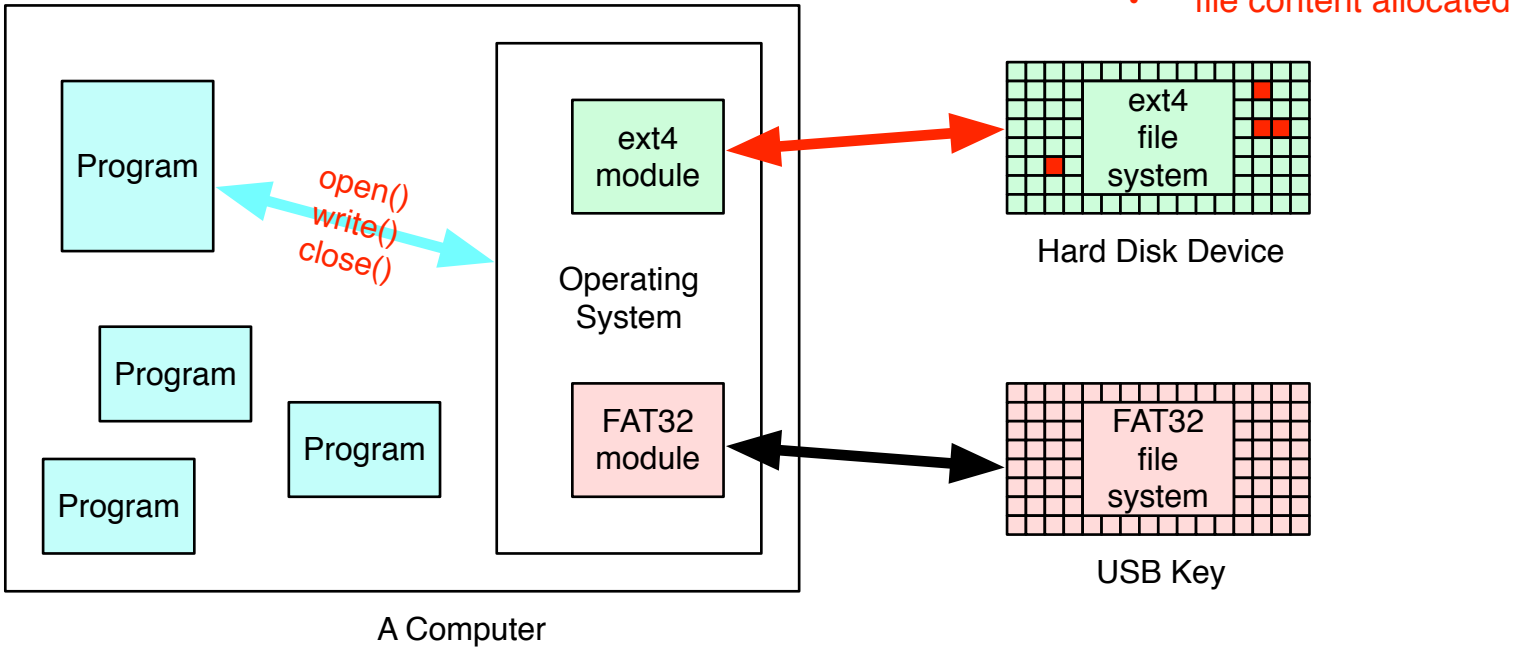
```c
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char *argv[]) {
    int fd;

    fd = open("hello.txt",O_CREAT|O_WRONLY,0700);
    write(fd,"hi!\n",4);
    close(fd);
}
```

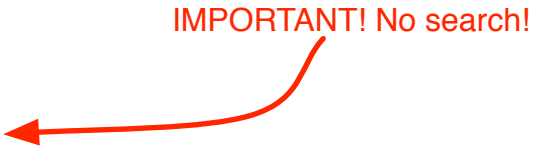
API for files 2

- dir content updated
- inode for file created
- file content allocated



API for directories

- opendir()
- readdir()
- closedir()
- rmdir()




```c
#include <stdio.h>
#include <unistd.h>
#include <dirent.h>

int main(int argc, char *argv[]) {
    DIR    *dh;
    struct dirent *entry;

    dh=opendir(".");
    while ((entry = readdir(dh)) != NULL) {
        printf("%lld %s\n",entry->d_ino,entry->d_name);
    }
    closedir(dh);
}
```

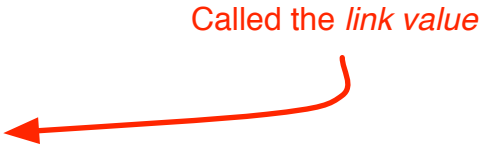
API for files or directories

- stat()
- rename()
- link()
- symlink()
- lstat()
- readlink()

... and others for changing inode properties

Symbolic links 1

The best way to understand a symbolic link is to imagine that it is...

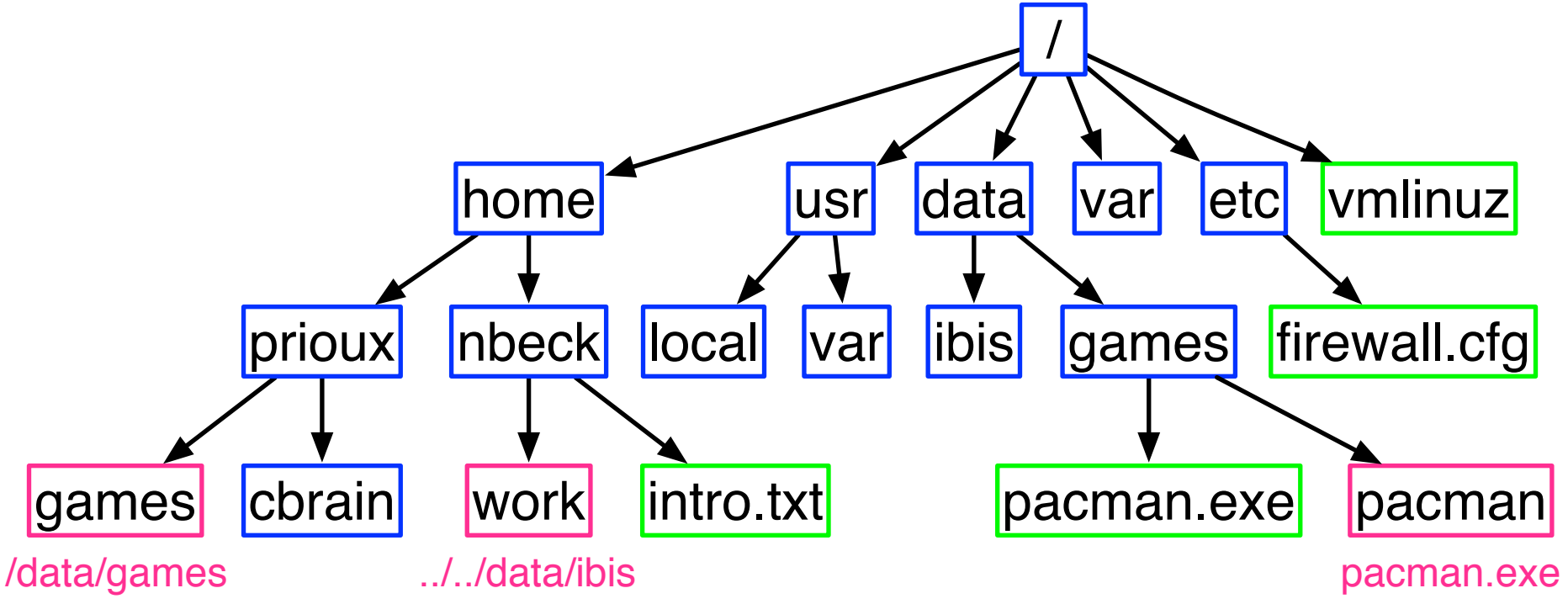


- a TEXT file...
- ... with a SINGLE line of text...
- ... which describe where to find another file or directory.

The link value can be...

- a relative path: "../../abc.txt" or "hi/how/do/you/do.txt"
- an absolute path: "/Volumes/Maps/Earth.pdf"

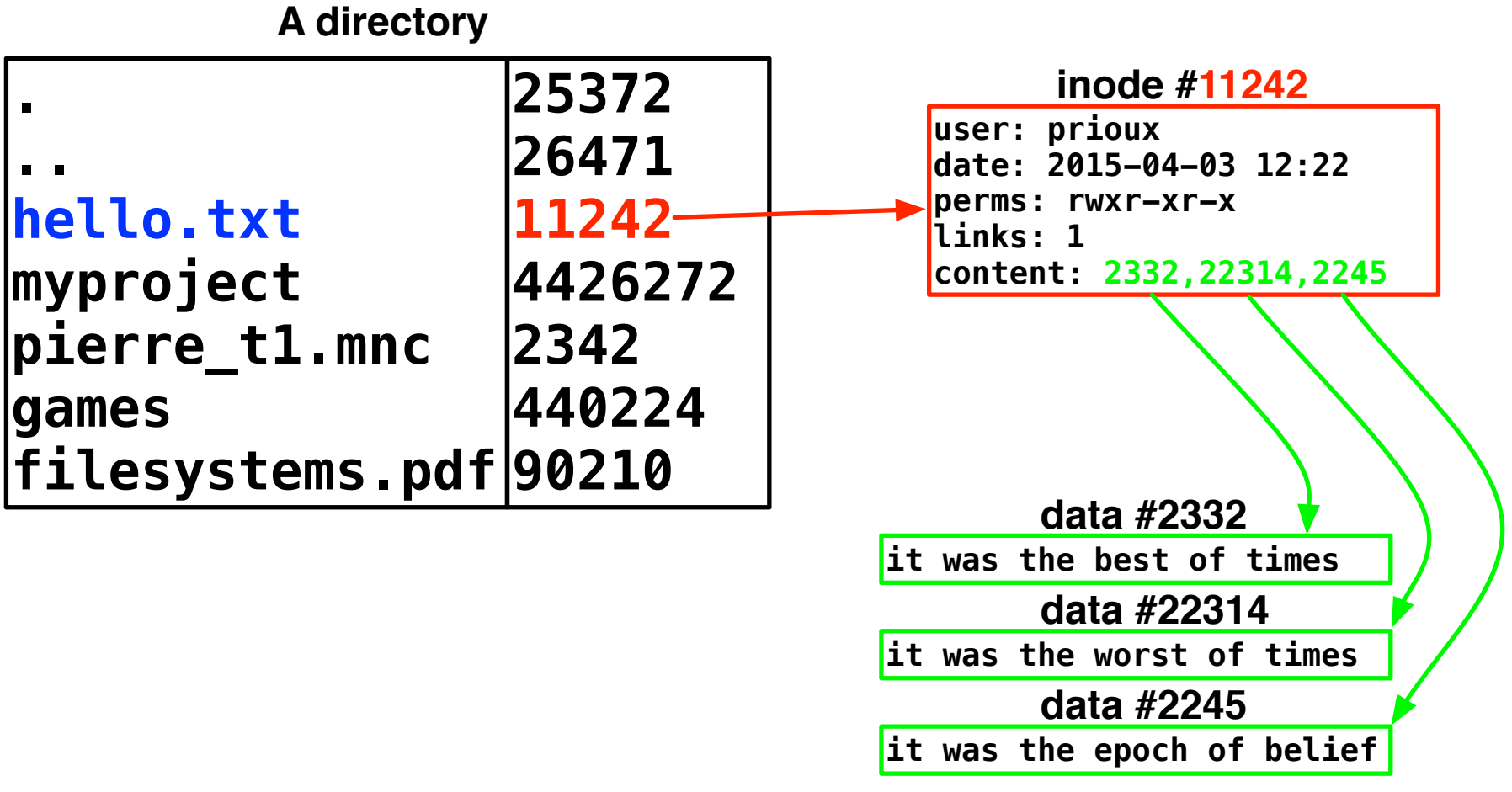
Symbolic links 2



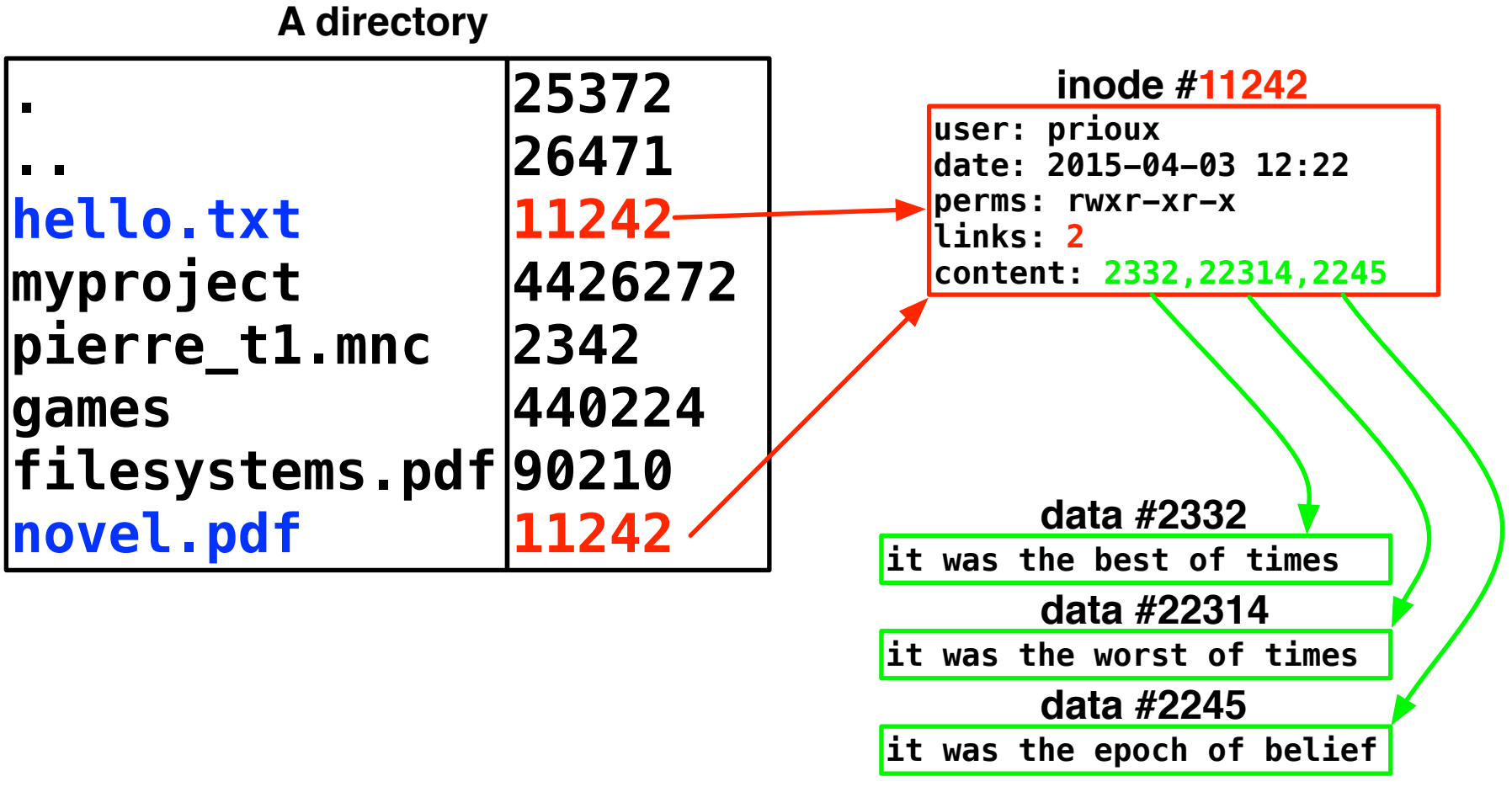
Notes:

- Not all file system types support symbolic links.
- Symbolic links can point to stuff on other filesystems.
- The link value can be anything and might not even point to anything that exist!

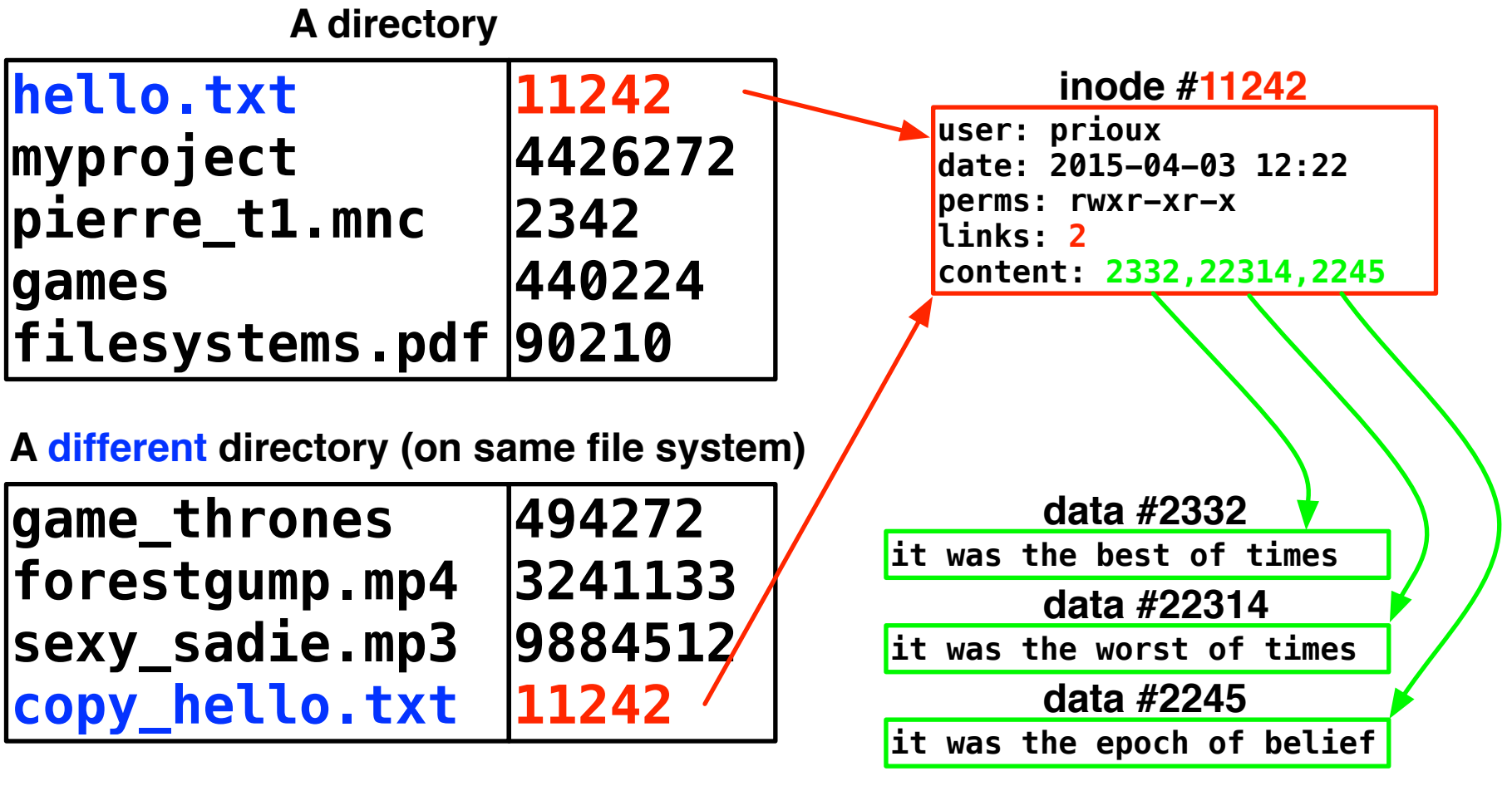
Hard links 1


A directory
.
..
hello.txt
myproject
pierre_t1.mnc
games
filesystems.pdf
25372
26471
11242
4426272
2342
440224
90210
inode #11242
user: prioux
date: 2015-04-03 12:22
perms: rwxr-xr-x
links: 1
content: 2332,22314,2245
data #2332
it was the best of times
data #22314
it was the worst of times
data #2245
it was the epoch of belief

Hard links 2


A directory
.
..
hello.txt
myproject
pierre_t1.mnc
games
filesystems.pdf
novel.pdf
25372
26471
11242
4426272
2342
440224
90210
11242
inode #11242
user: prioux
date: 2015-04-03 12:22
perms: rwxr-xr-x
links: 2
content: 2332,22314,2245
data #2332
it was the best of times
data #22314
it was the worst of times
data #2245
it was the epoch of belief

Hard links 3


A directory
hello.txt 11242
myproject 4426272
pierre_t1.mnc 2342
games 440224
filesystems.pdf 90210
A different directory (on same file system)
game_thrones 494272
forestgump.mp4 3241133
sexy_sadie.mp3 9884512
copy_hello.txt 11242
inode #11242
user: prioux
date: 2015-04-03 12:22
perms: rwxr-xr-x
links: 2
content: 2332,22314,2245
data #2332
it was the best of times
data #22314
it was the worst of times
data #2245
it was the epoch of belief

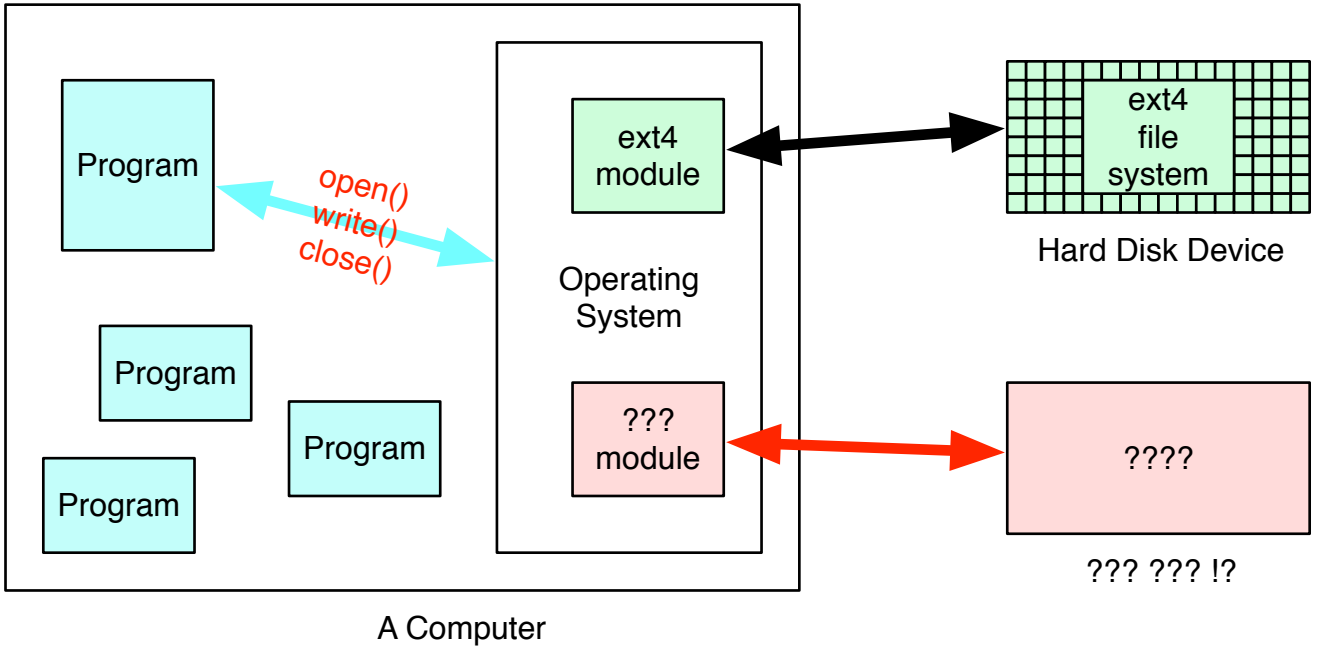
Hard links 4

- Hard links can only be created between contents on the SAME filesystem.
- None of the directory entries is more special than the others.
- The content and the inode is removed only when the last link is deleted.
- Not all file system types allow hard links.
- Modifying the content modifies it for all links!
- Hard links are NOT permitted for directories; the exceptions are "." and ".." (but done by OS).

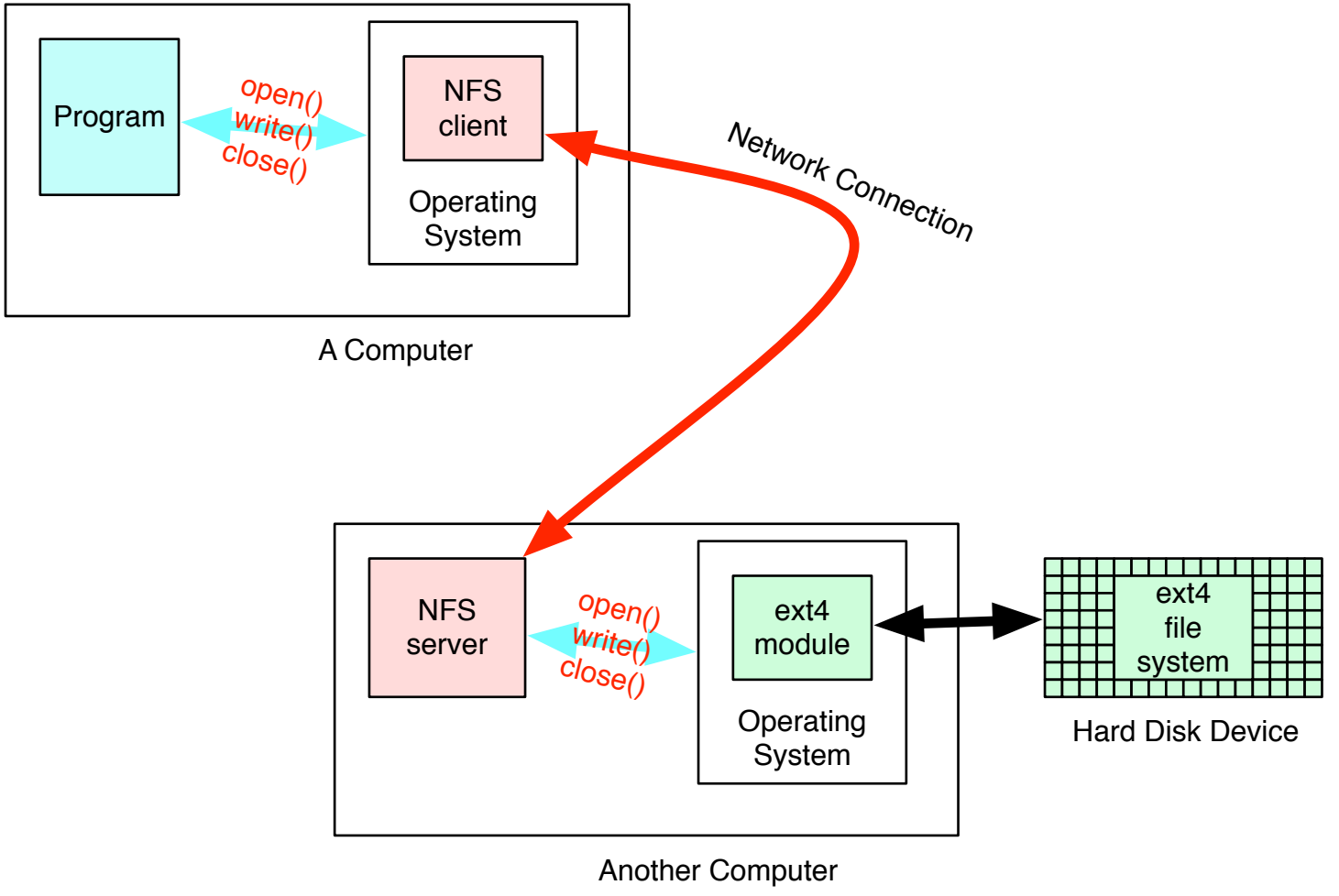
Links Demo

```
Bianca home Demo >
Bianca home Demo > echo "some content" > hello.txt
Bianca home Demo > ls -l
total 8
-rw-r--r--  1 prioux  staff  13 Apr 23 14:14 hello.txt
Bianca home Demo > ln -s hello.txt sym_to_hello.txt
Bianca home Demo > ls -l
total 16
-rw-r--r--  1 prioux  staff  13 Apr 23 14:14 hello.txt
lrwxr-xr-x  1 prioux  staff   9 Apr 23 14:14 sym_to_hello.txt@ -> hello.txt
Bianca home Demo > ln hello.txt hardlink_hello.txt
Bianca home Demo > ls -l
total 24
-rw-r--r--  2 prioux  staff  13 Apr 23 14:14 hardlink_hello.txt
-rw-r--r--  2 prioux  staff  13 Apr 23 14:14 hello.txt
lrwxr-xr-x  1 prioux  staff   9 Apr 23 14:14 sym_to_hello.txt@ -> hello.txt
Bianca home Demo > echo "more content" >> hello.txt
Bianca home Demo > ls -l
total 24
-rw-r--r--  2 prioux  staff  26 Apr 23 14:15 hardlink_hello.txt
-rw-r--r--  2 prioux  staff  26 Apr 23 14:15 hello.txt
lrwxr-xr-x  1 prioux  staff   9 Apr 23 14:14 sym_to_hello.txt@ -> hello.txt
Bianca home Demo > cat sym_to_hello.txt
some content
more content
Bianca home Demo > rm hello.txt
remove hello.txt? y
Bianca home Demo > cat sym_to_hello.txt
cat: sym_to_hello.txt: No such file or directory
Bianca home Demo > ls -l
total 16
-rw-r--r--  1 prioux  staff  26 Apr 23 14:15 hardlink_hello.txt
lrwxr-xr-x  1 prioux  staff   9 Apr 23 14:14 sym_to_hello.txt@ -> hello.txt
```

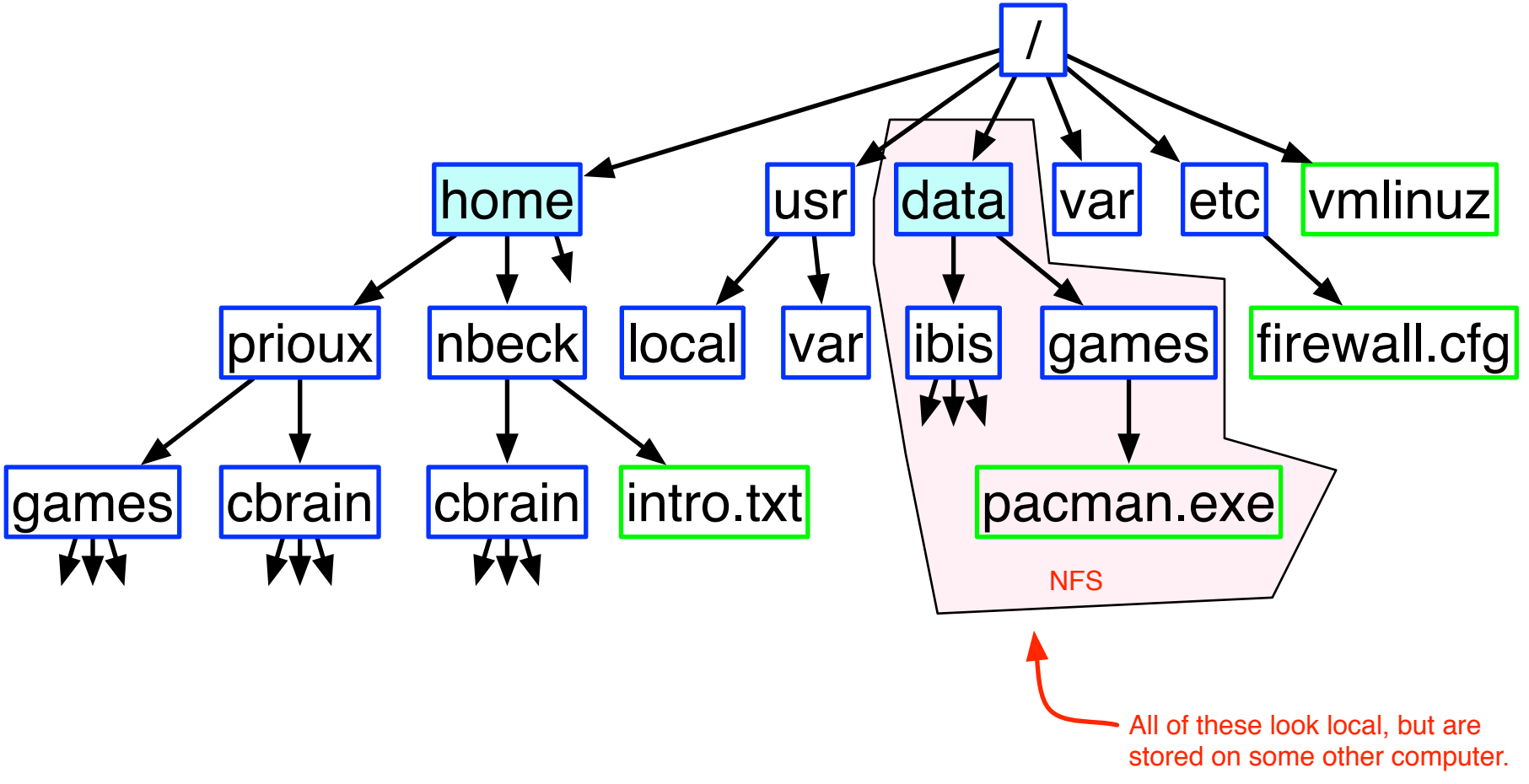
API abstraction



NFS 1


Program
open()
write()
close()
NFS
client
Operating
System
A Computer
Network Connection
NFS
server
open()
write()
close()
ext4
module
Operating
System
ext4
file
system
Hard Disk Device
Another Computer

NFS 2


/
home
usr
data
var
etc
vmlinuz
prioux
nbeck
local
var
ibis
games
firewall.cfg
games
cbrain
cbrain
intro.txt
pacman.exe
NFS
All of these look local, but are
stored on some other computer.

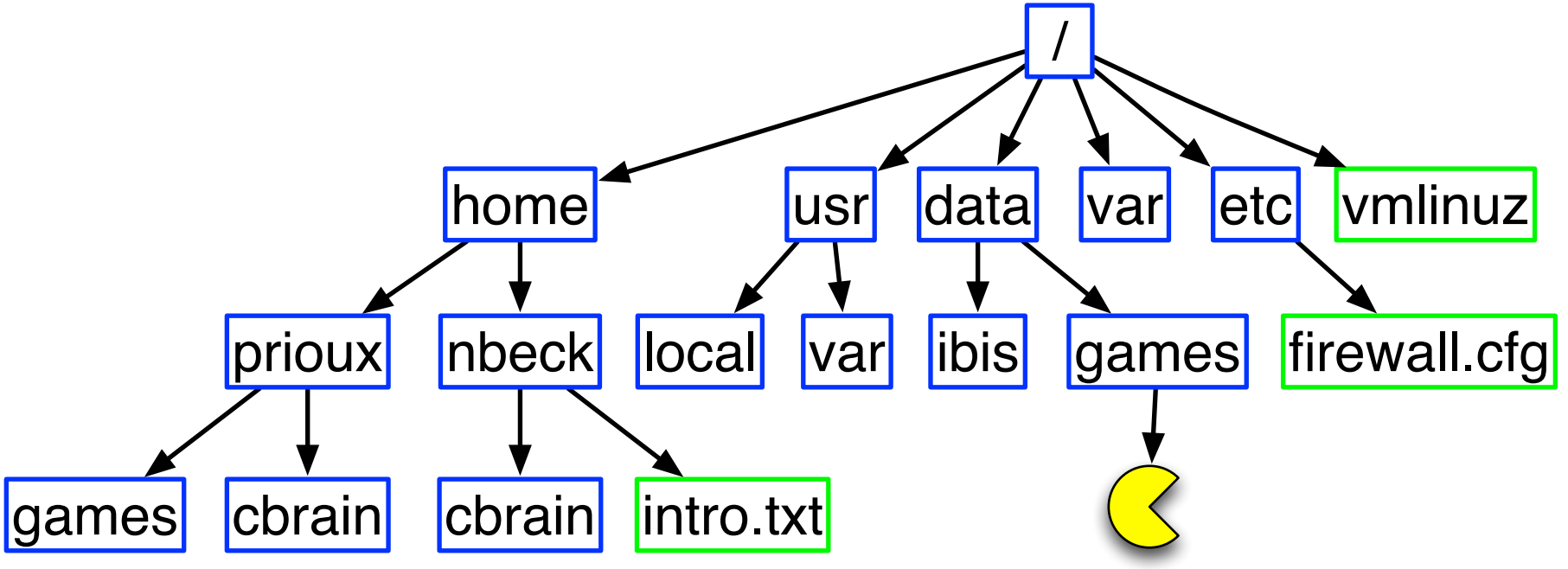
Operations that are FAST within a file system

- Renaming an entry (file or directory) in place.
- Creating a hard link.
- Creating a symbolic link.
- Unlinking a file with more than one link.
- Erasing files, as long as they are small.
- Creating directory entries in directories with few entries.
- Modifying inode information.

Operations that are SLOW within a file system

- Anything that involves manipulating a directory entry in a directory with already a large number of entries.
- Erasing large files.
- Resolving symbolic link spaghettis.

The END


/
home
usr
data
var
etc
vmlinuz
prioux
nbeck
local
var
ibis
games
firewall.cfg
games
cbrain
cbrain
intro.txt