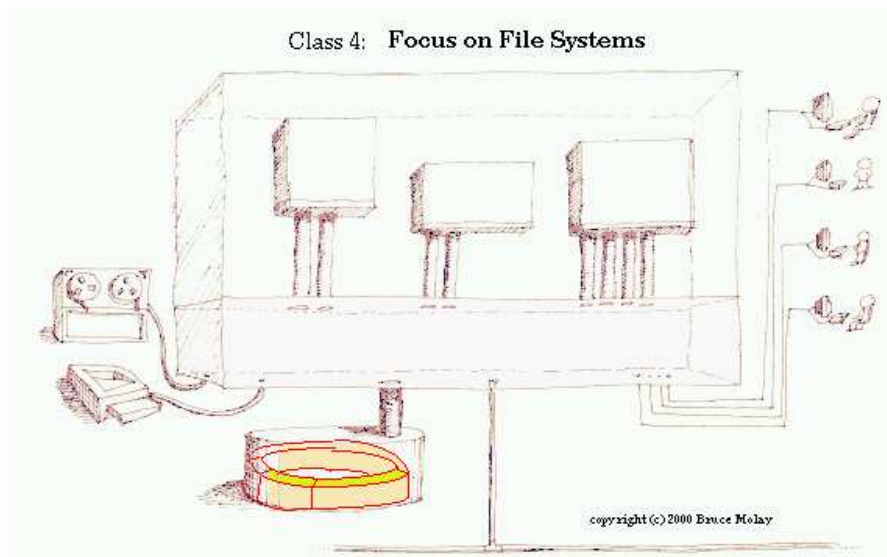


page000.html

000



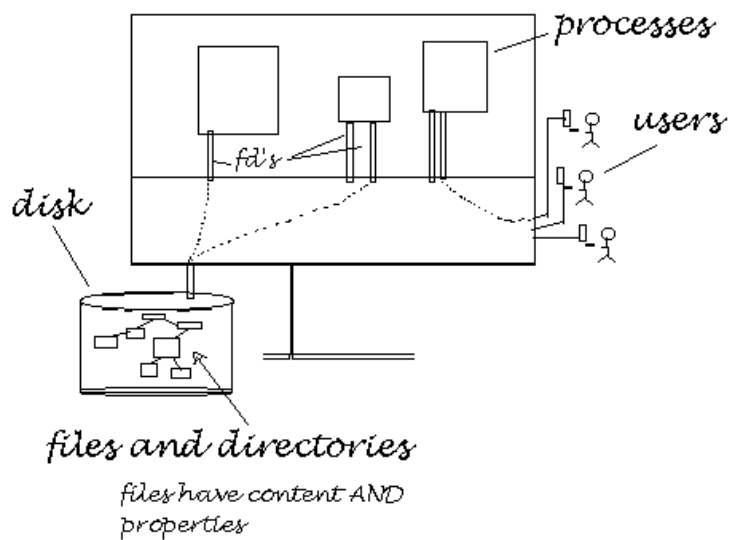
page001.html

Directory Trees and Disks

001

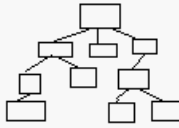
4) Focus on File Systems

The story so far



Tonight: focus on file systems

What users see: one big tree of dirs and files



The tree contains files, file info, and directories

The reality: one or more stacks of magnetic disks or memory chips.



How can multiple stacks of disks be made to look like a single tree of directories and files?

- What is the internal structure of a 'file system'?
- How do multiple disks merge into one tree?

project: write `pwd`

2) But first, a brief interruption to discuss error handling

`errno`
error code

`perror(char *)`
prints a message

`strerror(errno)`
returns a string

Fact: system calls return `-1` on error

But: what *is* the specific error?

example: `open()` can fail for many reasons

Q: How can the program determine which error it is?

A: The kernel sets a global value, known as `errno`, to a code in `/usr/include/errno.h`

Q2: How can a program tell the user what is the cause of the error?

A2: `perror(str)` prints "str: description"

Q3: What to do?

A: it depends

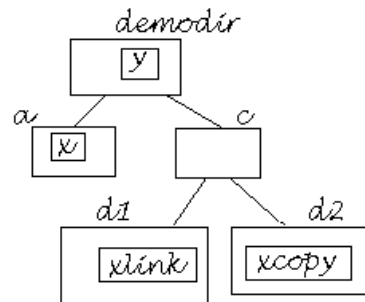
`perror_demo.c`

3) User view and Commands for file system

first, let's become familiar with the major fs-related commands by building a demo directory tree

commands

mkdir	cat
rmdir	cp
cd	rm
pwd	mv
	ln
du	
ls -la	
ls -R	



4) Almost no limits to tree structure

- *directories can contain lots of files*
- *directory depths can exceed the capacity of most fs commands*

Ex. dig
 while true
 do
 mkdir s
 cd s
done

What does du do with this directory structure? Or tree ls -lR, find, ... ?

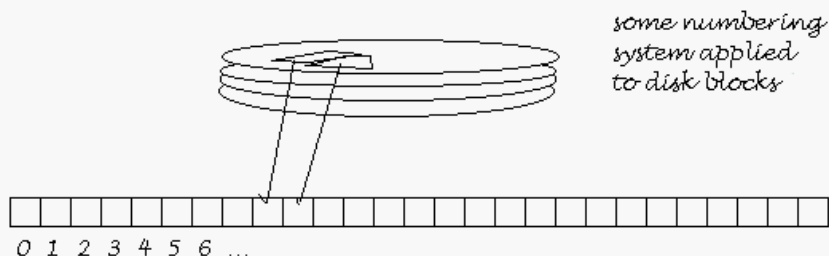
Note: some system maintenance tools might fail on these.

see also: dig_verbose



5) Face Reality: a disk is not a tree
we are not 'in' a directory

6) A disk is a stack of magnetic platters, disks, tracks, sectors, viewable as a sequence of DISK BLOCKS

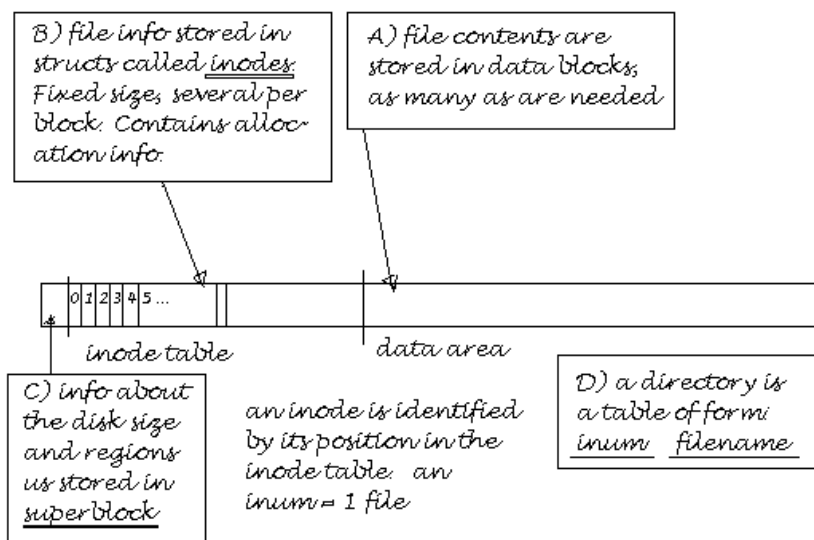


some numbering
 system applied
 to disk blocks

A 'disk' can also be a USB flash drive, the storage on a telephone, or storage device that consists of a sequence of blocks.

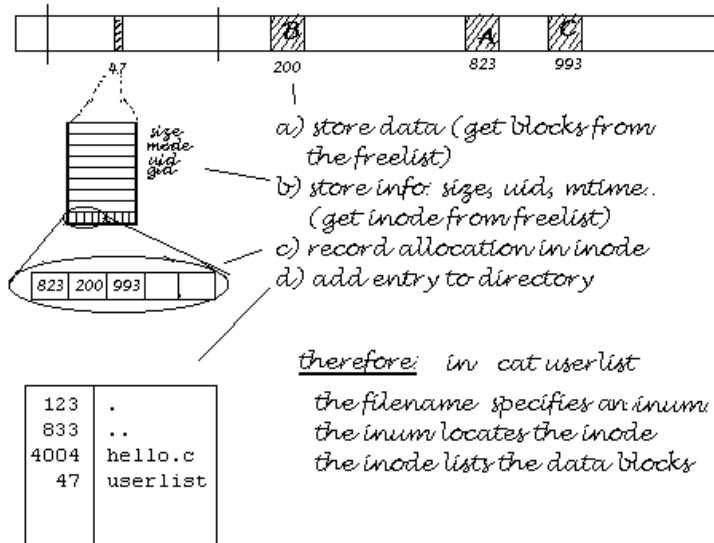
7) How can a numbered sequence of blocks
store files, file info, and directories ?

Ans: divide the disk into three regions, structure them sp:



8) Using this model to understand creating a file: e.g. `who > userlist`

say the file requires three disk blocks of storage...



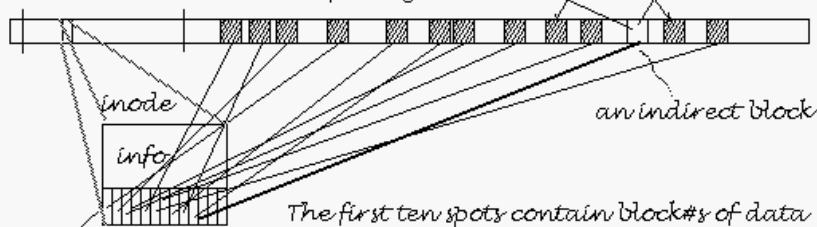
Details of recording disk allocation

fact: a large file is stored in many blocks

fact: the inode holds the info about which blocks contain the data

problem: how can a fixed-sized inode store a *long* block list?

ex: a file requiring 12 blocks



array of 13
block nums

note: a 'large'
file requires
more blocks
than size/blksiz

note: small directories
and symlinks can be
stored in the inode

The first ten spots contain block#s of data

block 11 contains the block # of a block of
block#s of data

block 12 contains the block # of a block of
block#s that contain block#s of data

block 13 contains the block # of a block
of block#s that contain block#s of block#s
that contain data

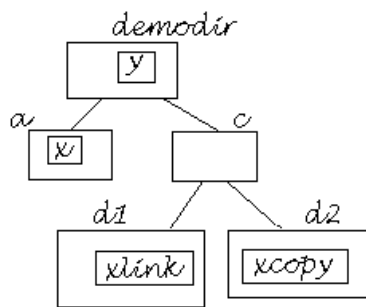
page010.html

010

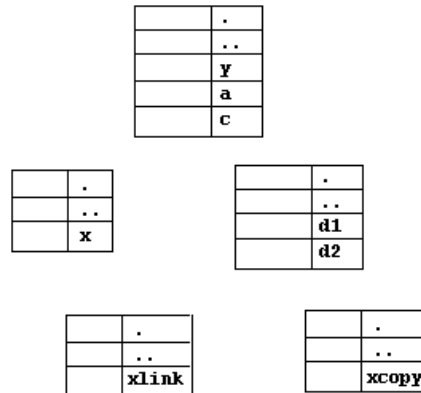
9) Using this model to revisit demodir

now that we know the internal structure of `dirs`, we can see what is **really** going on in our `demo dir` structure

user view



system view

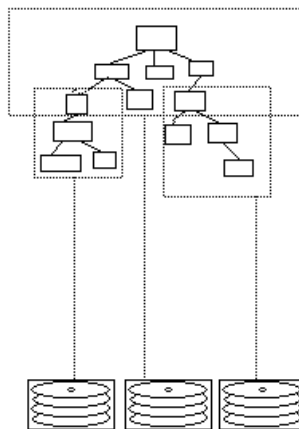


page011.html

Multiple disks, one tree

011

Question2: How can many disks form one tree?



Ans: Each disk contains a tree structure. That tree can be 'mounted' on another tree at any directory.

`stat()` info includes the inode *and* the device. That pair uniquely identifies the file.

Each file on a disk has an inode number.

Each disk has a device ID.

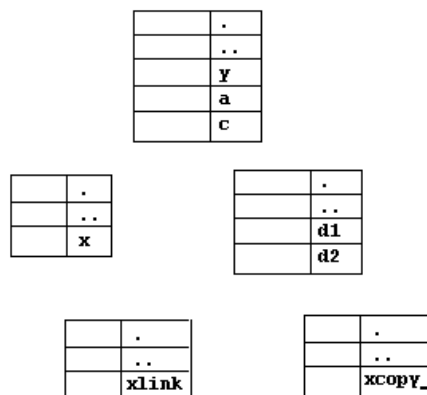
The 'mount' command reports what disks are attached to the tree at what directories. The directories are called 'mount-points'.

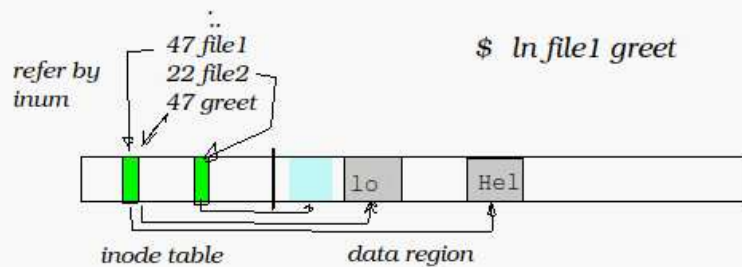
10) The System Calls for standard file ops

<i>command</i>	<i>syscall</i>	<i>action</i>
<i>rm</i>	<i>unlink()</i>	<i>removes a link if links == 0, deallocate</i>
<i>rmdir</i>	<i>rmdir()</i>	<i>delinks a directory</i>
<i>ln</i>	<i>link()</i>	<i>creates a new link</i>
<i>mv</i>	<i>link() then unlink() now rename()</i>	
<i>mkdir</i>	<i>mkdir()</i>	<i>creates new directory</i>

11) writing pwd

pwd prints the path to the current directory. But, the current directory only knows itself as "." How can its location in the tree be determined?



Hard Links: Different Names for the Same File

One file: two dir entries (i.e. two links)

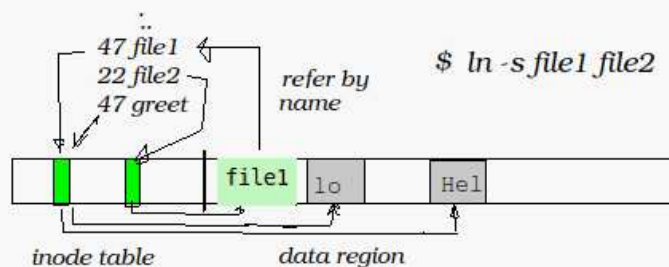
You can use either name. `chmod o+x file1`, `chmod u-r greet`

Q: What if you `rm file1`? What happens to `greet`?

A:

Note: In a city, each street has its own list of house numbers
Each file system has its own inode table (inums)

But: A hard link cannot refer to an inode on a different
file system. 123 Main St vs 123 Maple St

Symbolic Links: Another Way for Different Names, Same File

There are two files : the symlink is a file that contains
the name of its referent

You can use either name : `cat file1`, `cat file2`

Q: What if you `rm file1`? What happens to `file2` ?

A:

But: A sym link can contain the name of a file/dir
on a different filesystem.

page016.html

016

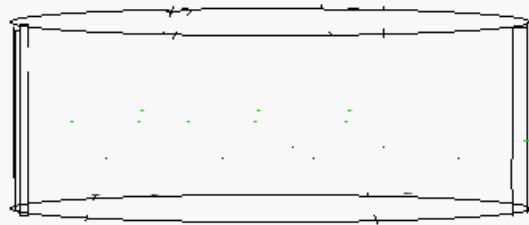
Making Links: Time Lapse Images

\$

A rectangular box representing a directory entry. It contains the text '83 .' on the first line and '74 ..' on the second line. There are small dots and lines inside the box, suggesting a list of files and their corresponding inodes and permissions.

83 .
74 ..

contents:

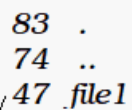


page017.html

017

Making Links: Time Lapse Images

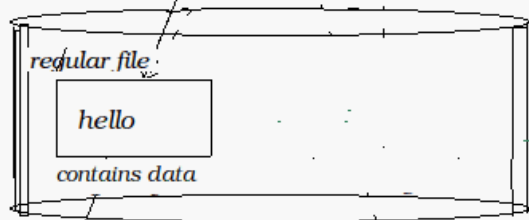
\$ echo hello > file1

A rectangular box representing a directory entry. It contains the text '83 .' on the first line, '74 ..' on the second line, and '47 file1' on the third line. There are small dots and lines inside the box, suggesting a list of files and their corresponding inodes and permissions.

83 .
74 ..
47 file1

create a file
called file1
containing
data: hello

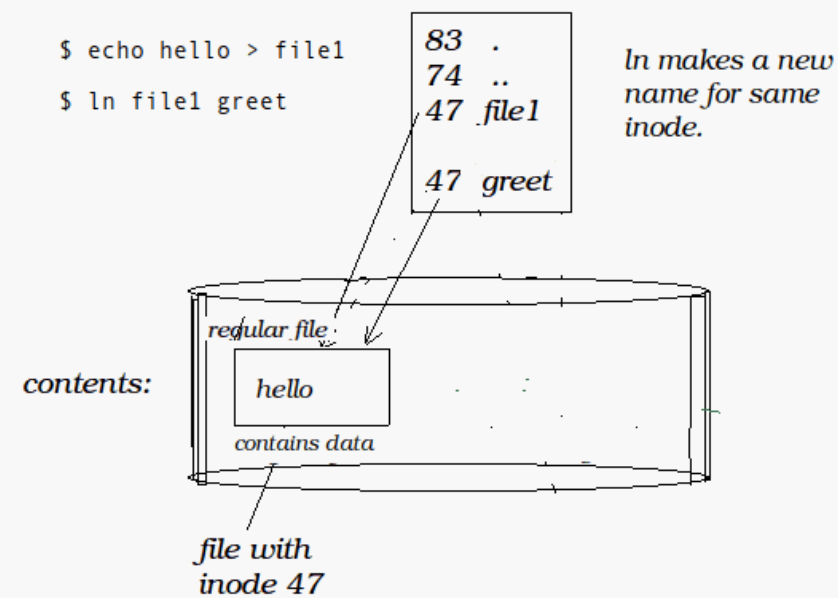
contents:



regular file:
hello
contains data
file with
inode 47

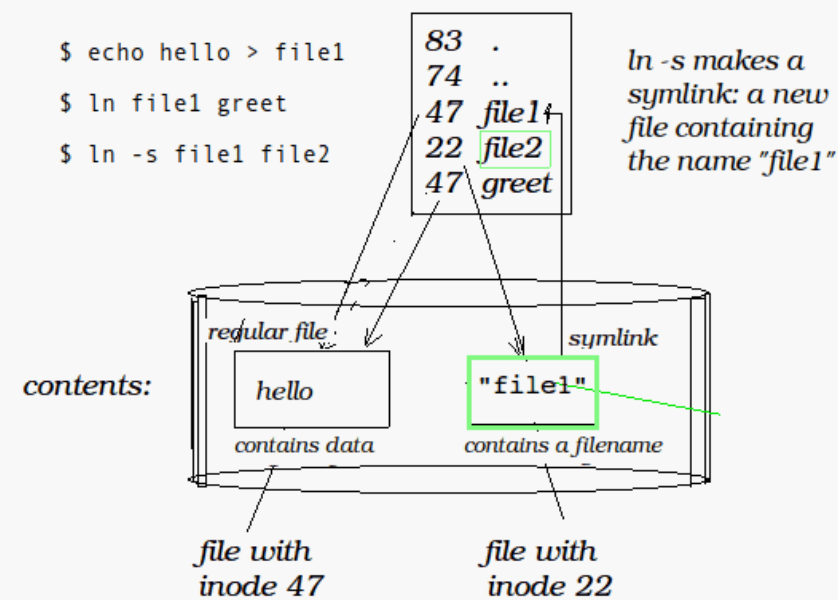
page018.html

018

Making Links: Time Lapse Images

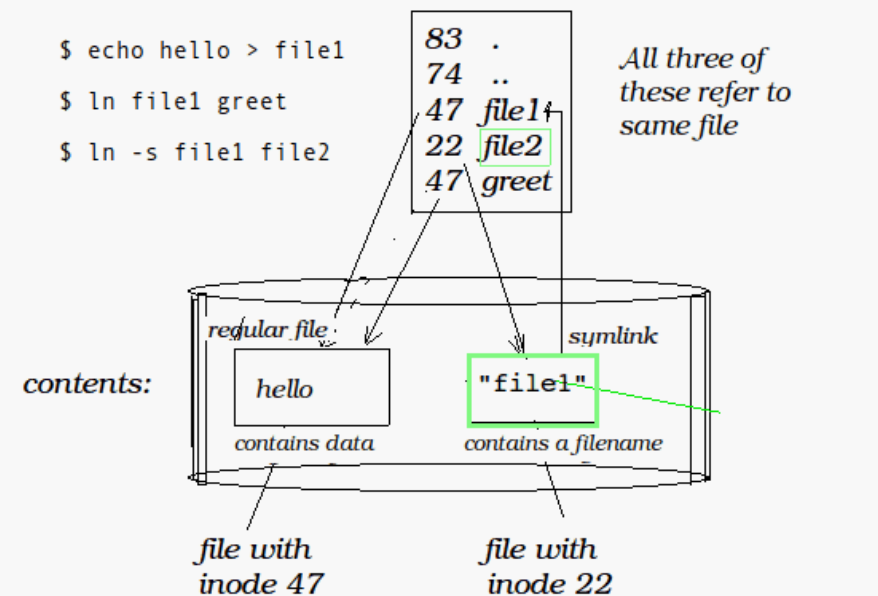
page019.html

019

Making Links: Time Lapse Images

page020.html

020

Making Links: Time Lapse Images

page021.html

021

Multiple Names for Same File: Two different ways to referHard Links

a directory entry that contains a name and inum.

Refers by inum

all three of these refer to the hello file

```

83 .
74 ..
47 file1
22 file2
47 greet

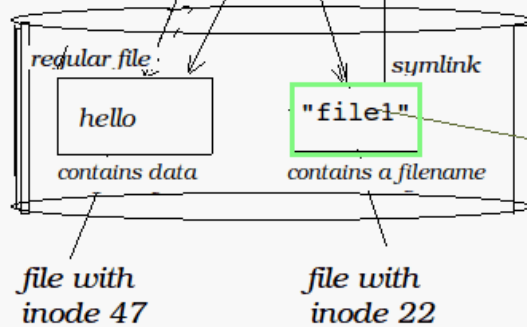
```

dir: link to 83
dir: link to 74
file: link to 47
symlink: link to 22
another link to 47

Symbolic Link:

a file that contains the name of a file:
Refers by name

contents:



Symlinks can refer to directories and files on other disks
Hard links cannot refer to directories or files on other disks

page022.html

022

Making Links: Time Lapse Images

\$

83	.
74	..

contents: