

**Basic Abstraction: FILE** (regular file)

- array of bytes, has low-level name (i.e., a number), can create/read/write/grow/delete, persistent

**Basic Abstraction: DIRECTORY** (special type of file)

- array of records, maps human-understandable file/dir name to low-level file/dir name, can create/delete/read

## **\*\* FILE OPERATIONS \*\***

**CREATION** - creat.c - look at it, run it, strace it

**ACCESS** (read) -

concept of **FILE DESCRIPTOR** - open files are tracked (in table)  
(not shared with other processes, in general)

**file descriptor**: per-process int to refer to given open file  
tracked per open file: **CURRENT OFFSET** (inc'd after read/write)

cat file1 -- strace cat file1

make file bigger w/ "echo a >> file1" in loop 5000x (**HOW BIG?**)

strace cat file1 > /dev/null (make sure to TRACK CURRENT OFFSET)

**GROW** (write) [FILE grow.c]

**how do you grow a file?** lseek() to seek to end of the file

**TRUNCATE** (make file smaller) [FILE trunc.c]

strace trunc, ls -al file1, truncate() syscall (not always zero)

**DELETE** - **ASK how to delete a file?** which system call? MYSTERY!

strace rm file1, show that it is "unlink"! (WHY??)

**RENAME** - **ASK how to rename a file?** -- mv file1 file2

strace it, see that rename() system call is used

**STAT** - **ASK how to get info about file?** - strace stat file2

which leads to:: **ASK what info should FS store about a file?**

all information is stored in per-file structure called an **inode**

**KEY**: low-level name of file is thus the "inode number"

(short for **index node**, because original unix FS inodes in array)

Some other interesting things in there:

SIZE (and BLOCKS), OWNERSHIP, PERMISSIONS, ACCESS TIMES,

POINTERS to BLOCKS (we can't see these)

## **\*\* DIRECTORY OPERATIONS \*\***

**Directory:** just a **SPECIAL TYPE of FILE**

can create/read/delete (but NOT write to, directly)

stores mapping human-readable name->low-level name (inode#)

**CREATE/REMOVE** - strace mkdir foo, strace rmdir foo

**READDIR** - can READ as well - readdir.c

(show how to build this using opendir(), readdir(), closedir())

**ASK: what program is this??** (answer: it's "ls")

Some **SPECIAL** directories:

**ROOT:** "/" at top of tree - **CURRENT:** "." - **PARENT:** ".."

### **PATHNAMES**

**ABSOLUTE:** /x/y/z -- root / dir / dir / ... / file or dir

**RELATIVE:** foo/bar.c -- dir / dir / ... / file or dir

key notion: **SEPARATOR** (/ is the right one, others used)

each process has **CURRENT WORKING DIRECTORY** (getcwd())

**WELL-KNOWN INODE # FOR ROOT** -> ls -ali / -> root is 2

**PATH TRAVERSAL** - Revisiting OPEN (What does OPEN() do?)

fd = open("/a/b/c/file1", O\_RDONLY);

start @ ROOT (or CWD if relative), **TRAVERSE** root directory,  
'a', 'b', 'c', finally finds 'file1' and reads its inode

fd is then used to refer to 'file structure' for this file

### **LINKS (aka HARD LINKS)**

NOT a special type of file (but you might think it is)

how to create? -> ln file1 file2 -> strace it! (link(!))

eg: cat file1 - cat file2 - ls -ali file\* -> same inode #

stat file1 file2 -> see LINK COUNT

rm file1 - cat file2 -> **WILL IT WORK?**

stat file2 -> see LINK COUNT

rm file2

**And now you know: why REMOVE is done by UNLINKing a file**

each unlink removes single reference to file (**REF COUNT--**)

when final unlink occurs, file is removed from file system

**LINKS (SYMBOLIC aka SOFT)** - a special file type

ln -s file1 file2 -> strace to show symlink() system call

ls -al file1 file2 -> SHOW HOW IT IS A SPECIAL FILE TYPE

NOTE FILE SIZE of file2

**(it is 5 bytes: can anyone guess why?)**

rm file1 - cat file2 --> dangling pointer!

why needed? (to point to other FSes, dirs in cycle)

## **\*\* MOUNTING FILE SYSTEMS \*\***

each file system is stored in a **"VOLUME"**

looks like a disk (but could be part of a disk, a partition)

but all stitched together into **SINGLE TREE** of all file systems

type **"mount"**

show root directory

show how another disk is mounted on **"/scratch.1"**

can use this to assemble full file-system tree!

(instead of having c: and d: and e: on desktop)