Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`
- The OS reads the block containing the root directory off disk and scans through it for the name `home`

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`
- The OS reads the block containing the root directory off disk and scans through it for the name `home`
- It finds it and gets the inode number for `home`

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`
- The OS reads the block containing the root directory off disk and scans through it for the name `home`
- It finds it and gets the inode number for `home`
- It reads the inode off disk and finds it refers to a directory

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`
- The OS reads the block containing the root directory off disk and scans through it for the name `home`
- It finds it and gets the inode number for `home`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk

Filesystems

Inodes

When a program opens a file, the OS must find where on disk the file lives

Say we are looking for the file `prog.c` with a cwd of `/home/rjb`

- The name is incomplete, so the OS prepends the cwd giving `/home/rjb/prog.c`
- The OS reads the block containing the root directory off disk and scans through it for the name `home`
- It finds it and gets the inode number for `home`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `rjb`

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`
- It finds it and gets the inode number for `prog.c`

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`
- It finds it and gets the inode number for `prog.c`
- It reads the inode off disk and finds it refers to a file

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`
- It finds it and gets the inode number for `prog.c`
- It reads the inode off disk and finds it refers to a file
- It reads the blocks containing the file off disk

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`
- It finds it and gets the inode number for `prog.c`
- It reads the inode off disk and finds it refers to a file
- It reads the blocks containing the file off disk

This must be done for every file opened

Filesystems

Inodes

- It finds it and gets the inode number for `rjb`
- It reads the inode off disk and finds it refers to a directory
- It reads the block containing the directory off disk
- It scans the directory for the name `prog.c`
- It finds it and gets the inode number for `prog.c`
- It reads the inode off disk and finds it refers to a file
- It reads the blocks containing the file off disk

This must be done for every file opened

Again, caching can be used to great effect: keeping copies of the inodes and directories in memory, rather than re-reading them every time

Filesystems

Inodes

If we want more than one filesystem on a disk, or more than one kind of filesystem, we can split the disk into separate *partitions*

Filesystems

Inodes

If we want more than one filesystem on a disk, or more than one kind of filesystem, we can split the disk into separate *partitions*

A partition is just a chunk of disk owned by a single filesystem

Filesystems

Inodes

If we want more than one filesystem on a disk, or more than one kind of filesystem, we can split the disk into separate *partitions*

A partition is just a chunk of disk owned by a single filesystem

So we can have multiple filesystems on a single disk, e.g., two Unix filesystems and a Windows filesystem

Filesystems

Inodes

If we want more than one filesystem on a disk, or more than one kind of filesystem, we can split the disk into separate *partitions*

A partition is just a chunk of disk owned by a single filesystem

So we can have multiple filesystems on a single disk, e.g., two Unix filesystems and a Windows filesystem

Each filesystem has its own inode tables (or whatever it requires) and are logically quite separate

Filesystems

Inodes

Note that inode 23 on one partition is different to inode 23 on another partition, meaning we can't have hard links across filesystems

Filesystems

Inodes

Note that inode 23 on one partition is different to inode 23 on another partition, meaning we can't have hard links across filesystems

We *can* have soft links across filesystems, as soft links are by names, not inode numbers: this is really why soft links were invented

Filesystems

Mounting

Under Unix, a filesystem can be *mounted* on another filesystem

Filesystems

Mounting

Under Unix, a filesystem can be *mounted* on another filesystem

The name comes from when disks needed to be physically mounted on the drives by system operators

Filesystems

Mounting

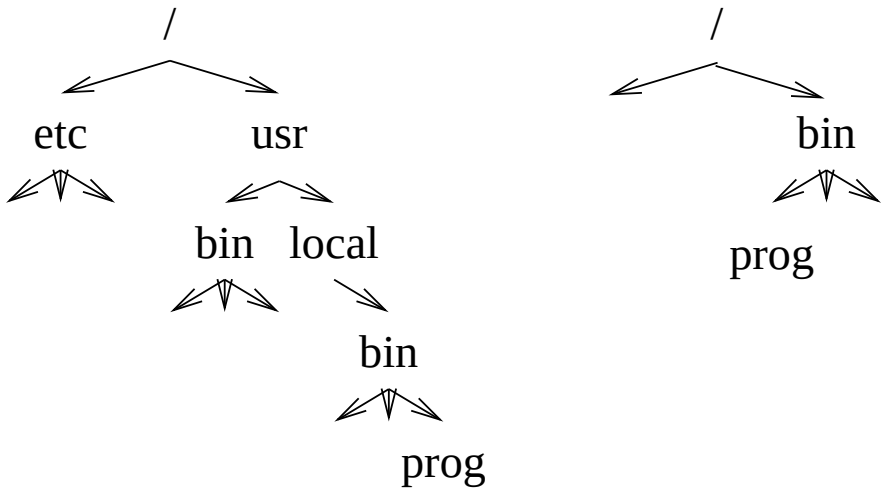
Under Unix, a filesystem can be *mounted* on another filesystem

The name comes from when disks needed to be physically mounted on the drives by system operators

A *mount point* is a special inode that says: “now go and look at this filesystem”

Filesystems

Mounting

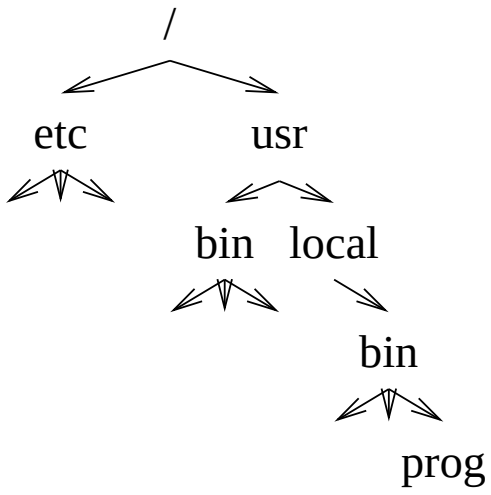


Filesystem A Filesystem B

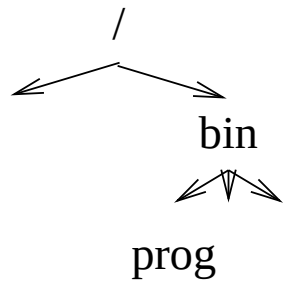
Filesystems A and B exist separately, maybe on separate disks, with filesystem A as the system root

Filesystems

Mounting



Filesystem A

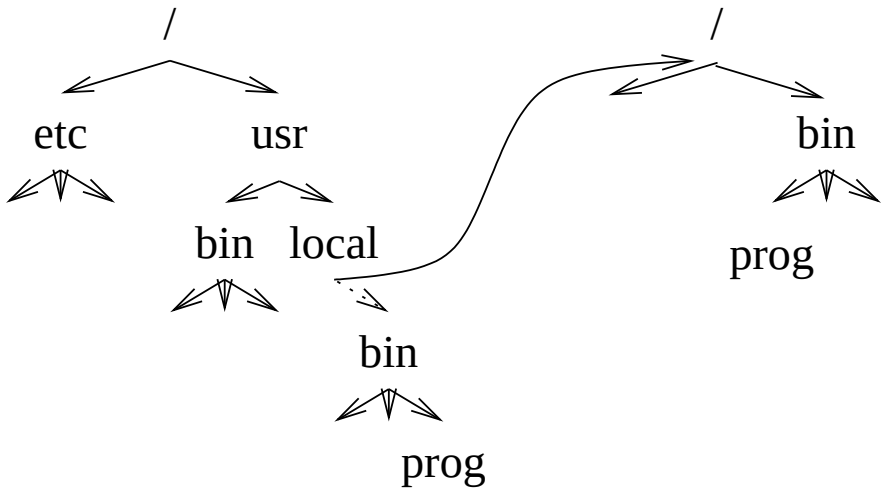


Filesystem B

Filename `/usr/local/bin/prog` refers to the `prog` on A

Filesystems

Mounting

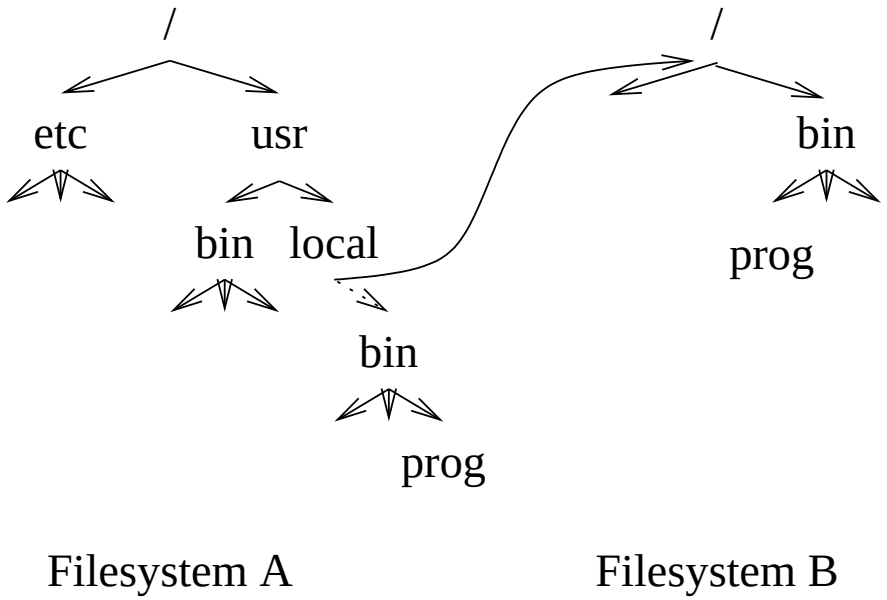


Filesystem A Filesystem B

If we mount filesystem B at the *mount point* `/usr/local` this *hides* the part of the hierarchy below `local`

Filesystems

Mounting



Filesystem A Filesystem B

And now name `/usr/local/bin/prog` refers to the `prog` on B

Filesystems

Mounting

When the file lookup reads the inode for the mount point at `/usr/local` it switches filesystem and continues looking from the root of B

Filesystems

Mounting

When the file lookup reads the inode for the mount point at `/usr/local` it switches filesystem and continues looking from the root of B

This means that we can have many partitions presented as a single unified name space

Filesystems

Mounting

When the file lookup reads the inode for the mount point at `/usr/local` it switches filesystem and continues looking from the root of B

This means that we can have many partitions presented as a single unified name space

And partition B could be a separate disk; or on a USB key; or on a read-only medium like a CD

Filesystems

Mounting

Note that B will have its own inode table, so there can't be a hard link of, say, a name in `/usr/bin` to a name in `/usr/local/bin`

Filesystems

Mounting

Note that B will have its own inode table, so there can't be a hard link of, say, a name in `/usr/bin` to a name in `/usr/local/bin`

In fact, B might even have a completely different kind of filesystem, perhaps not based on inodes

Filesystems

Mounting

Note that B will have its own inode table, so there can't be a hard link of, say, a name in `/usr/bin` to a name in `/usr/local/bin`

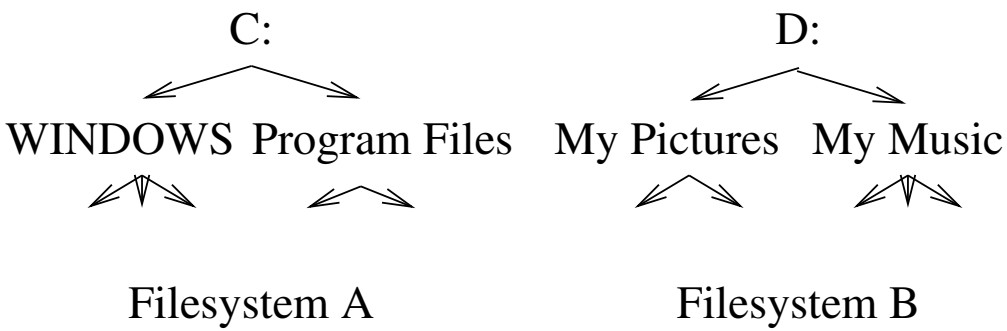
In fact, B might even have a completely different kind of filesystem, perhaps not based on inodes

Or can be on a separate machine if this was a mount of a *network* disk

Filesystems

Mounting

This is completely different from Windows where each partition is separate and has a prefix like `C:`



Filesystems

Mounting

Going the other way, mechanisms exist for gluing several disks together to make them appear as a single partition: this can be for making huge filesystems out of small disks, or for reliability through redundancy (RAID)

Filesystems

Other filesystems you might like to look at

- btrfs
- ext4
- FAT, VFAT
- FUSE
- GFS (Global File System)
- Google File System
- HFS+
- ISO 9660
- JFFS2
- Lustre
- NFS
- NTFS
- OCFS2
- procfs
- Reiser
- ReFS (Resilient File System)
- UnionFS
- ZFS

Filesystems

Other filesystems you might like to look at

- btrfs
- ext4
- FAT, VFAT
- FUSE
- GFS (Global File System)
- Google File System
- HFS+
- ISO 9660
- JFFS2
- Lustre
- NFS
- NTFS
- OCFS2
- procfs
- Reiser
- ReFS (Resilient File System)
- UnionFS
- ZFS

Also see “List of file systems” on Wikipedia

Filesystems

Exercise. Solid state disks (SSDs) are common these days. What differences do they bring to the way filesystems should be implemented?

Exercise. Read about the various kinds of RAID filesystems and the benefits they bring

Filesystems

Dilbert
by Scott Adams

□