CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Directories

When accessing secondary storage, users do not work directly with, e.g., disk

File system is organization of secondary storage into meaningful units for users

File systems provide two main abstractions:

- files (data to be stored)
- directories (organization)

Need some way of organizing files

Typically do not access single `server.c` for entire system, but instead access

`/home/student/os-course/project-1/server.c`

`server.c` is the file; other parts are directories

Directory is map between filename and "directory entry"

Directory entry tells system where file is located

Directory entry may include file location and metadata directly, or may be pointer to structure elsewhere that has required information

In Linux, file location and metadata stored in inodes

Directory, then, is map between filename and inode

Not necessarily the case in every system, though similar principles apply

Example directory in Linux

Name	inode
prog1.c	12063521
prog2.c	12063524
a.out	12063525

Directories must themselves be stored on secondary storage

In Linux, directory is just special type of file

/parent/

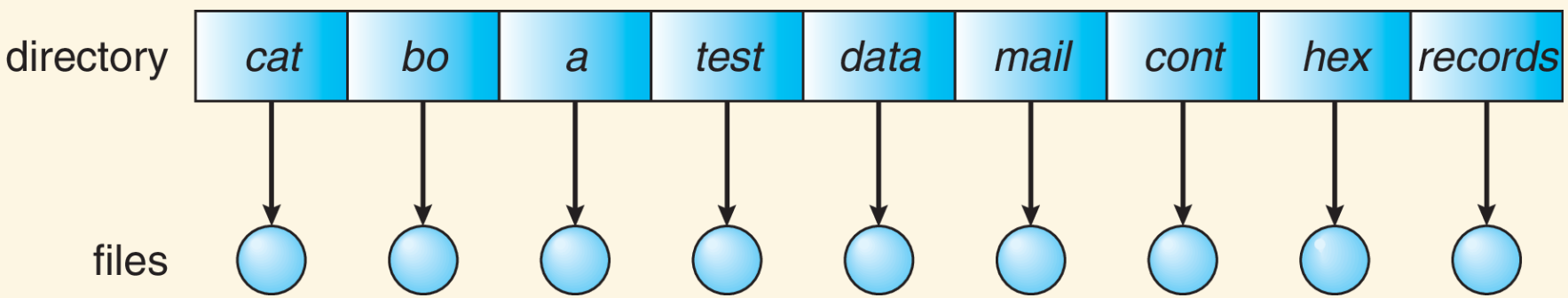
Name	inode
prog1.c	12063521
prog2.c	12063524
a.out	12063525
subdir	12063535

/parent/subdir/

Name	inode
helper.c	12063529
subsubdir	12063532

Single-level directory

If directories could not hold other directories, structure mapping filenames to inodes would be very simple



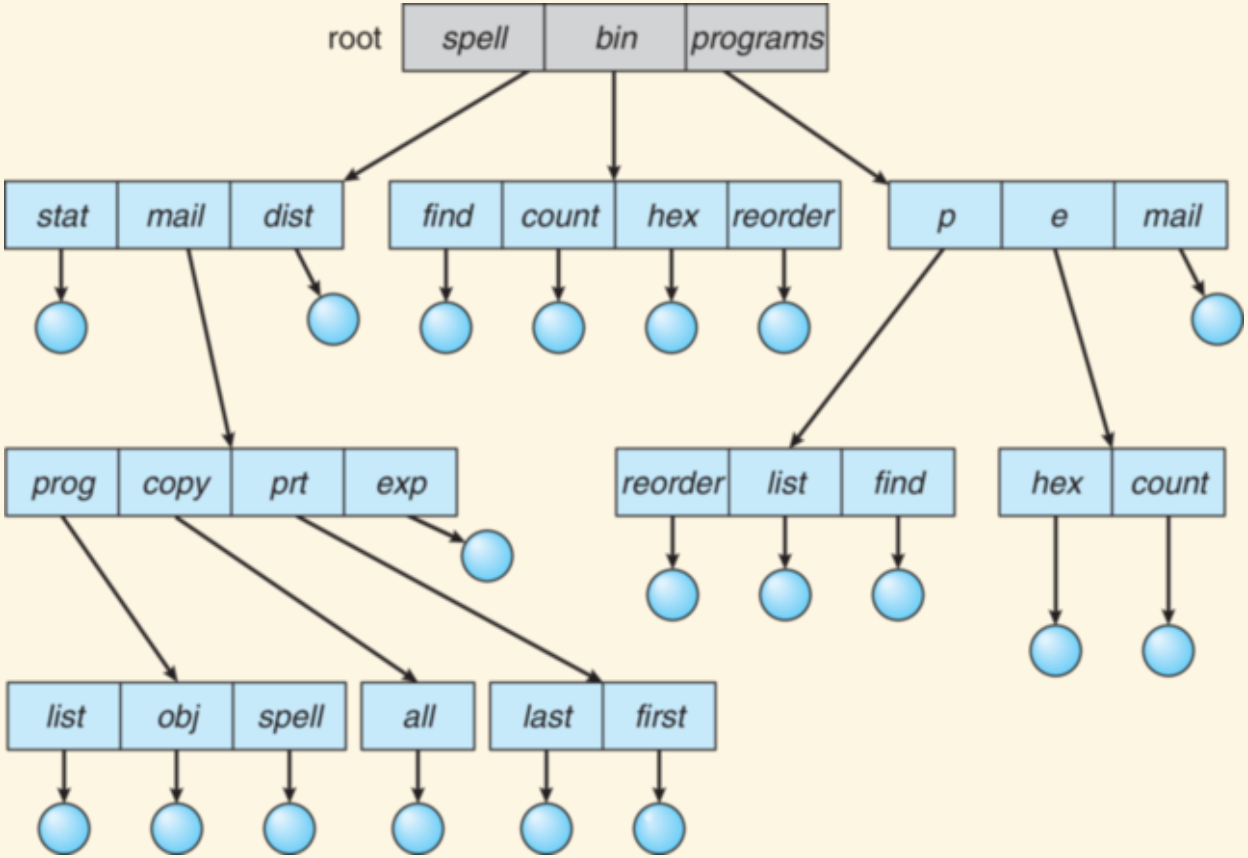
Does not allow for duplicate filenames

Two users cannot both have `project1.c`

Very simple conceptually, but not realistic

Tree-structured directories

When directories may point to files and other directories, tree structure emerges


root
spell
bin
programs
stat
mail
dist
find
count
hex
reorder
p
e
mail
prog
copy
prt
exp
reorder
list
find
hex
count
list
obj
spell
all
last
first

Must specify full **path name** to uniquely identify file

Path name is full path from root to file

`/programs/p/list` in previous diagram

Generally, each process has *current directory* that ideally contains most files of interest to process

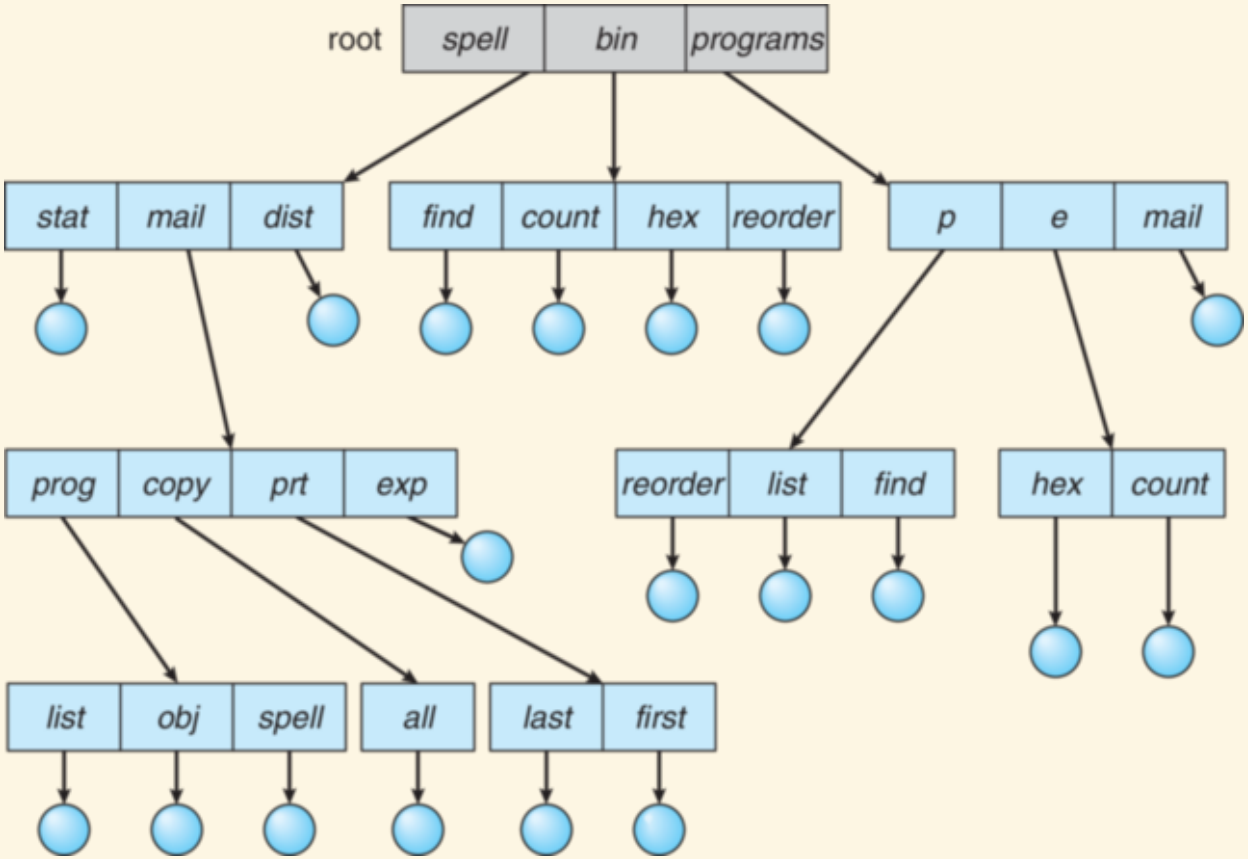
Not not use full path name to specify files in current directory

Absolute path name is path to file from root

Relative path name is path to file from current working directory

This is where Linux `cd` command comes from -- "change directory"

Adjusts which files can be accessed using relative path names



If working directory is `programs`, file `hex` can be accessed as either

- `/programs/e/hex` (absolute path)
- `e/hex` (relative path)

We will see that tree structure gets more complicated when we allow links