**System-Level I/O**

**Unix I/O**

- Implemented using exceptions/interrupts

- Two sets of I/O: system-level and C level

- Robust I/O: special wrappers using good coding practice

- Unix I/O functions accessed via system calls

- All I/O devices are represented as files; even the kernel is represented as a file

- A Linux file is a sequence of *m* bytes

- Elegant mapping of files to devices allows kernel to export simple Unix I/O interface

- Opening and closing files, reading and writing a file, changing the current file position

- Each file has a type indicating its role in the system

- Regular file: contains arbitrary data (video/audio/text/etc.)

- Directory: index for a related group of files

- Socket: communicates with a process on another machine

-Regular files can either be text files or binary files; kernel does not know the difference between the two

- A text file is a sequence of text lines; a text line is a sequence of characters terminated by newline char ‘\n’

- Directories consist of an array of links, where each link maps a filename to a file

- Directory contains at least two entries:

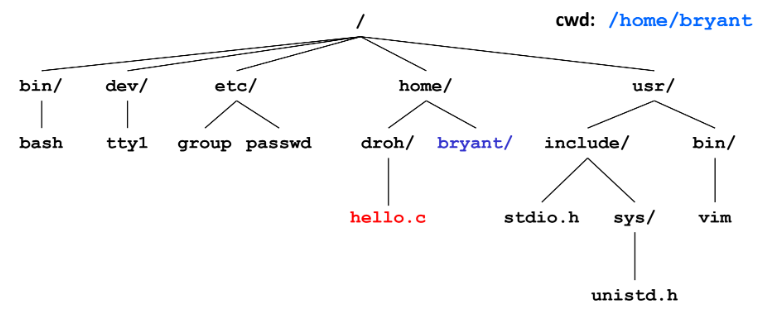
. (dot): link to itself

.. (dot dot) link to its parent

- Kernel maintains current working directory (cwd) for each process

- Absolute pathname starts with / and denotes path from root (/home/droh/hello.c)

- Relative pathname denotes path from current working directory (../droh/hello.c)



- Opening a file informs the kernel that you are getting ready to access that file

- Returns a small identifying integer file descriptor; fd == -1 indicates that an error occurred

- Each process created by Linux shell begins life with three open files associated with a terminal:

0: standard input (stdin)

1: standard output (stdout)

2: standard error (stderr)

- Closing a file informs the kernel that you are finished accessing that file (don’t close twice)

- Reading a file copies bytes from the current file position to memory and then updates file position

- Returns number of bytes read from file into buffer

- nbytes < 0 indicates an error

- Short counts (nbytes < sizeof (buf) ) are possible and are not errors

- Writing a file copies bytes from memory to the current file position and then updates current file position

- Returns number of bytes written from buffer to file

- nbutes < 0 indicates an error

- Short counts can occur when:

- encountering EOF on reads

- reading text lines from a terminal

- reading and writing network sockets

- Short counts never occur when:

- reading from disk files (except for EOF)

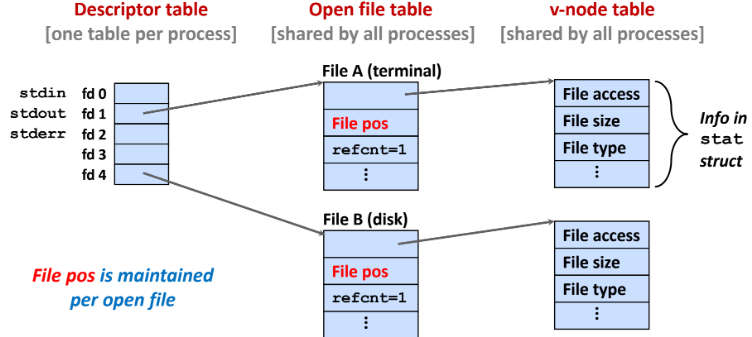
- writing to disk files

**Metadata, sharing, and redirection**

- Metadata is data about data; per-file metadata is maintained by kernel and is accessed by users with stat and fstat functions

- Unix kernel represents open files with two descriptors referencing two distinct open files

- Descriptor 1 (stdout) points to terminal, and descriptor 4 points to open disk file



- File sharing is two distinct descriptors sharing the same disk file through two distinct open file table entries

- Calling open twice with the same filename argument; different logical but same physical file

- Child process inherits its parent’s open files when fork is called

- refcnt is incremented

- Shell implements I/O redirection (ls > foo.txt) by copying the descriptor table entry oldfd to entry newfd

1. Open file to which stdout should be redirected

2. Call dup2

**Standard I/O**

- C standard library contains a collection of higher-level standard I/O functions

- Standard I/O models opens files as streams

- Abstraction for a file descriptor and a buffer in memory

- Applications often read/write one character at a time, but implementing this as Unix I/O calls is expensive

- Buffered read grabs a block of bytes and takes one byte at a time from buffer

- Standard I/O functions use buffered I/O

- Can see buffering in action with strace

**RIO (robust I/O) package**

- RIO is a set of wrappers that provide efficient and robust I/O in apps, such as network programs that are subject to short counts

- Two different kinds of functions

- Unbuffered input and output of binary data

- Buffered input of text lines and binary data

**Closing Remarks**

- Standard I/O and RIO are implemented using low-level UNIX I/O

- Generally, use the highest-level I/O functions you can