Unix File Descriptors and Pipes

Godmar Back

Virginia Tech

February 2, 2024

VIRGINIA TECH

Unix File Descriptors

- A file descriptor is a handle that allows user processes to refer to files, which are sequences of bytes
- Unix represents many different kernel abstractions as files to abstract I/O devices: e.g., disks, terminals, network sockets, IPC channels (pipes), etc.
- Provide a uniform API, no matter the kind of the underlying object
  - `read(2)`, `write(2)`, `close(2)`, `lseek(2)`, `dup2()`, and more
  - May maintain a read/write position if seekable
  - But note: not all operations work on all kinds of file descriptors
- Are represented using (small) integers obtain from system calls such as `open(2)`
- Are considered low-level I/O
- Are inherited/cloned by a child process upon fork()
- Are retained when a process exec()'s another program
- Are closed when a process exit()s or is killed

VIRGINIA TECH

Standard Streams

- By convention, 0, 1, 2 are used for standard input, standard output, and standard error streams
- Programs do not have to open any files; they are preconnected; thus programs can use them without needing any additional information
- Control programs (shell), or the program starting a program can set those up to refer to some regular file, terminal device, or something else
- When used, they access the underlying kernel object in the same way as if they'd open it themselves
- Programs should, in general, avoid changing their behavior depending on the specific type of object their standard streams are connected to
  - Exceptions exist, e.g., flushing strategy of C's stdio depends on whether standard output is a terminal or not
  - Python 2 `sys.stdout.encoding` fiasco

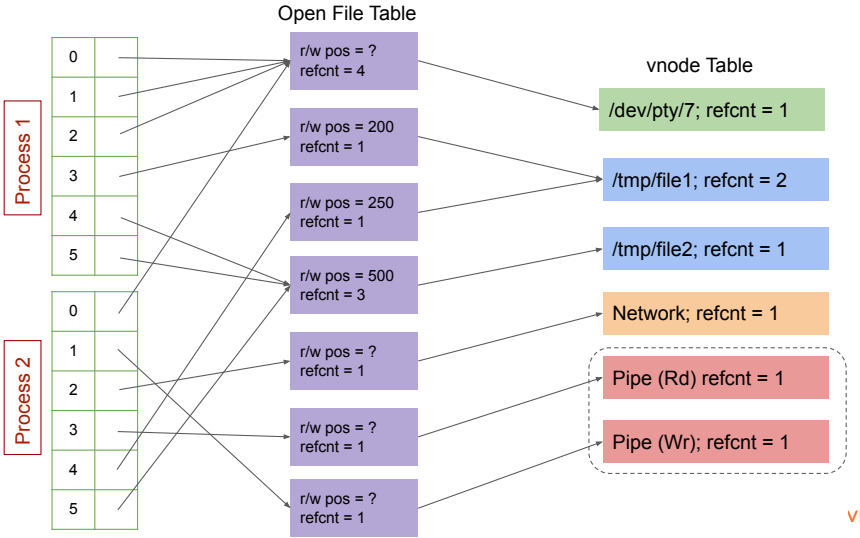
VIRGINIA TECH

File Descriptors – The subtle parts

- To properly understand file descriptors, must understand their implementation inside the kernel
- File descriptors use 2 layers of indirection, both of which involve reference counting
  - (integer) file descriptors in a per-process table point to entries in a global open file table
  - per-process file descriptor table has a limit on the number of entries
  - each open file table entry maintains a read/write offset (or position) for the file
  - entries in the open file table point to entries in a global "vnode" table, which contains specialized entries for each file-like object
- File descriptor tables are (generally) per-process, but processes can duplicate and rearrange entries

VIRGINIA TECH

In-Kernel Management of File Descriptors



VIRGINIA TECH

File Descriptor Manipulation

- `close(fd)`:
  - clear entry in file descriptor table, decrement refcount in open file table
  - if zero, deallocate entry in open file table and decrement refcount in vnode table
  - if zero, deallocate entry in vnode table and close underlying object
  - for certain objects (pipes, socket), closing the underlying object has important side effects that occur only if all file descriptors referring to it have been closed
- `dup(int fd)`: create a new file descriptor referring to the same open file table entry as file descriptor fd, increment refcount; returns lowest available (unused) file descriptor number
- `dup2(int fromfd, int tofd)`: if `tofd` is open, close it. Then, assign `tofd` to the same open file entry as `fromfd` (as in `dup()`, increment refcount
- On `fork()`, the child inherits a copy of the parent's file descriptor table (and the reference count of each open file table entries is incremented)
- On `exit()` (or abnormal termination), all entries are closed

VIRGINIA TECH

Pipes

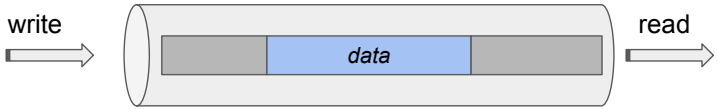


Figure 1: A Unix pipe is a FIFO, bounded buffer that provides the abstraction of a unidirectional stream of bytes flowing from writer to reader

- Writers:
  - can store data in the pipe as long as there is space
  - blocks if pipe is full until reader drains pipe
- Readers:
  - drains pipe by reading from it
  - if empty, blocks until writer writes data
- Pipes provide a classic "bounded buffer" abstraction that
  - is safe: no race conditions, no shared memory, handled by kernel
  - provides flow control that automatically controls relative progress: e.g., if writer is BLOCKED, but reader is READY, it'll be scheduled. And vice versa.
- Created unnamed; file descriptor table entry provide for automatic cleanup

VIRGINIA TECH