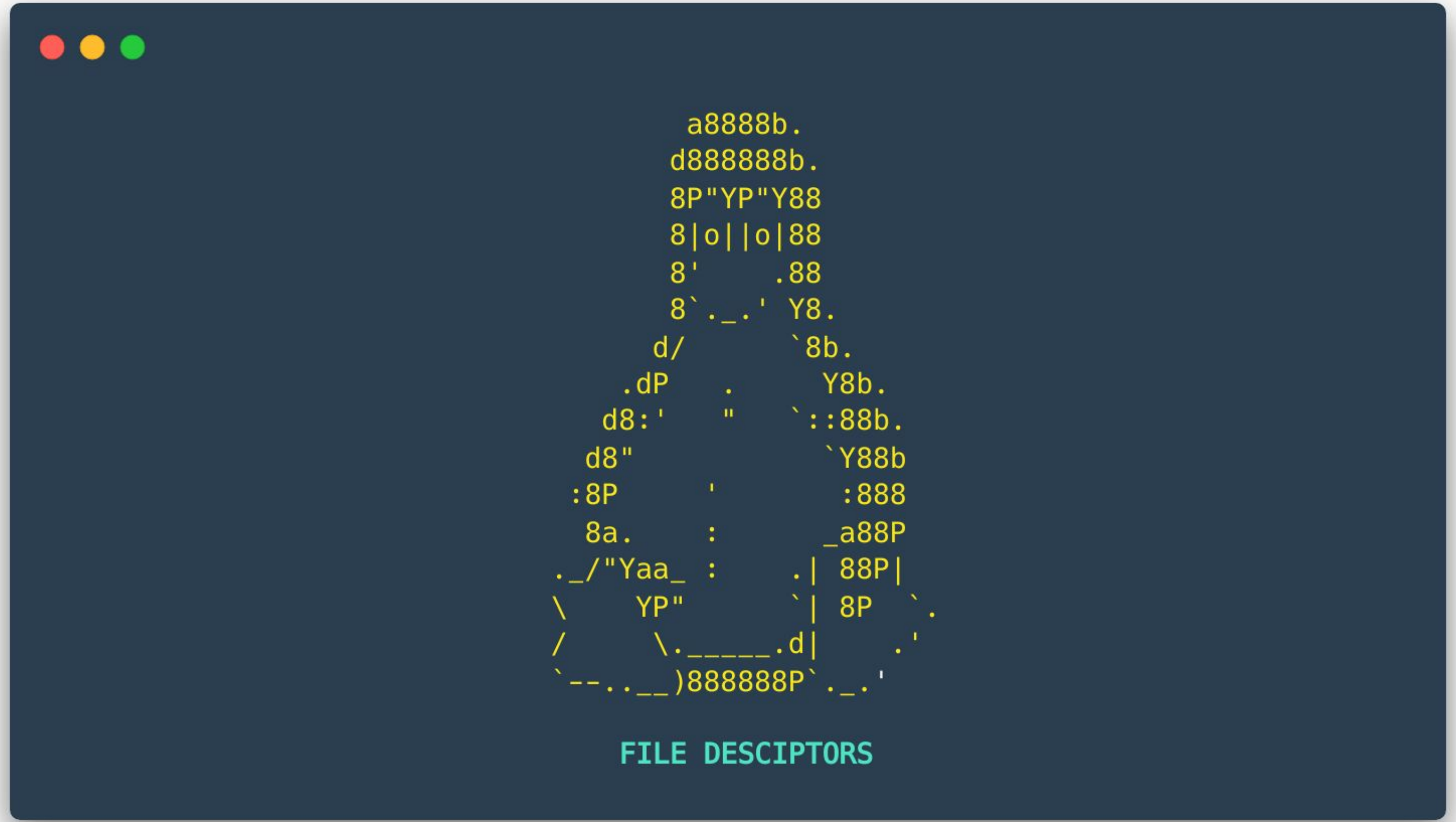
:')

Tutorial 3

File Descriptors, Pipes and Signals



```
        a8888b.
       d888888b.
       8P"YP"Y88
       8|o||o|88
       8'    .88
       8`._.' Y8.
      d/      `8b.
    .dP   .     Y8b.
   d8:'   "   `::88b.
  d8"           `Y88b
 :8P     '       :888
  8a.    :      _a88P
._/"Yaa_ :    .| 88P|
\    YP"      `| 8P  `.
/     \._____.d|    .'
`--..__)888888P`._.'
```

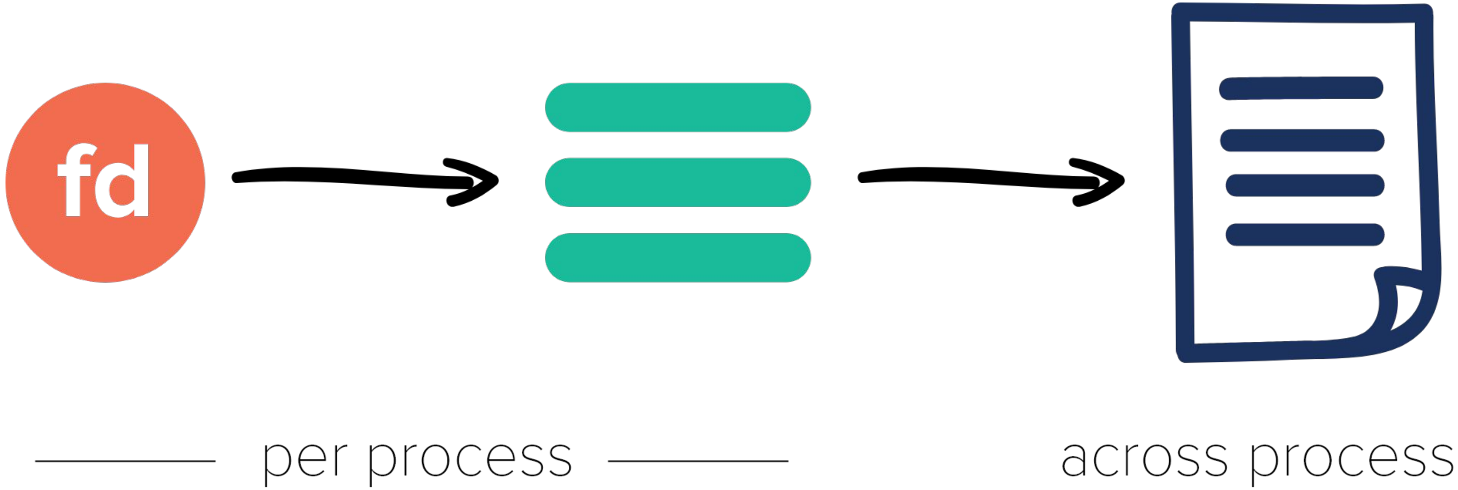
FILE DESCIPTORS

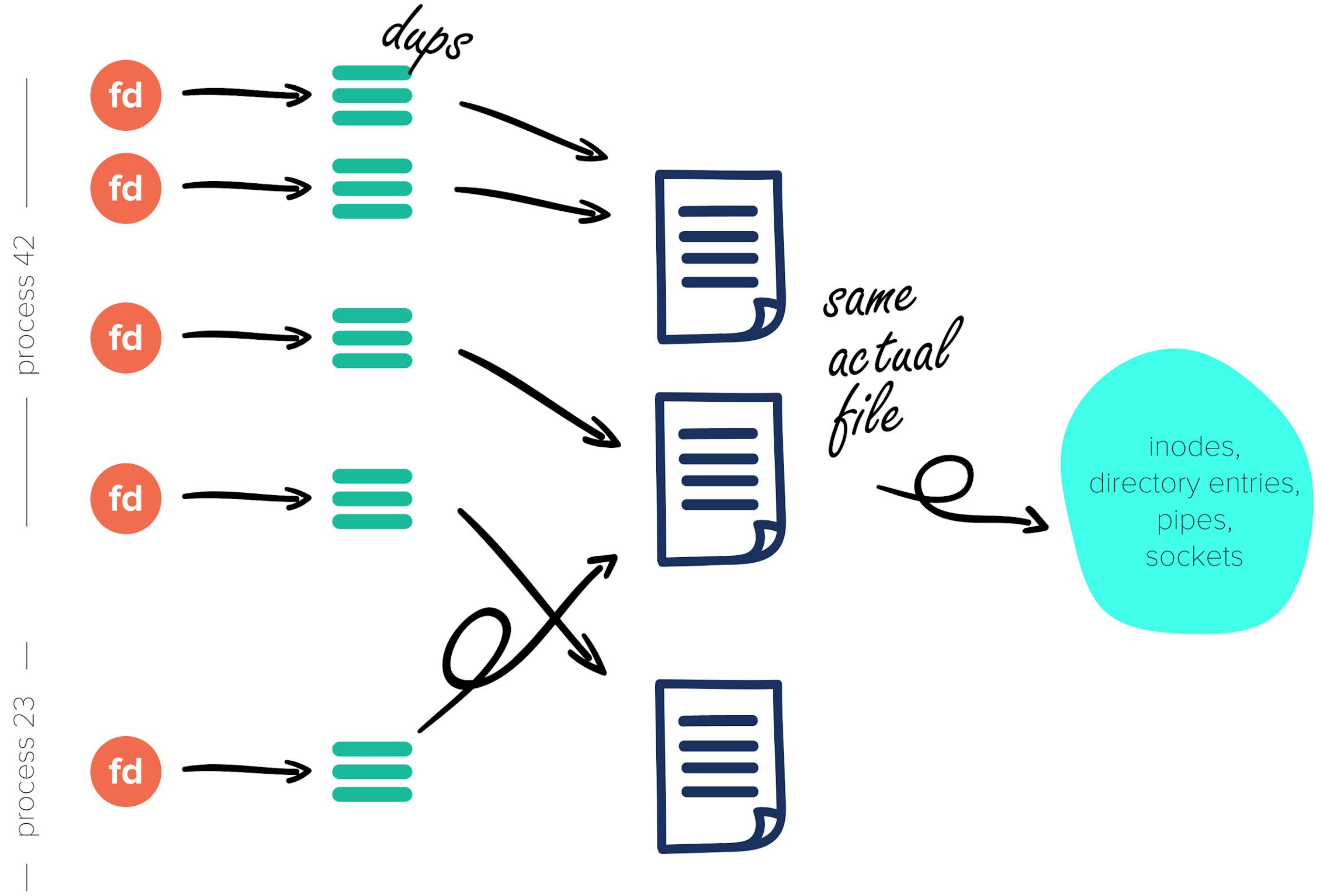
The File Descriptor

- A small non-negative integer.
- Designates an **open file** *in a process*.
- Seen a few of them before: `0` (`stdin`), 1 (`stdout`), 2 (`stderr`).
- The kernel keeps a table of open file descriptors for each process, mapping each to a **file descriptor structure** (`struct fd` in Linux).
- This structure contains a pointer to an **open file description**.

The Open File Description Structure

- `struct file` on Linux.
- There can be multiple file descriptors pointing (via the file descriptor structure) to the same open file description, from multiple processes.
- If the file descriptors (even belonging to different processes) are due to the same original `open` system call, they point to the same open file description.
- Created each time `open` is called; multiple calls to `open` even for the same file would create multiple open file descriptions.
- Stores the mode, seek position, pointers to the `inode`'s, etc.

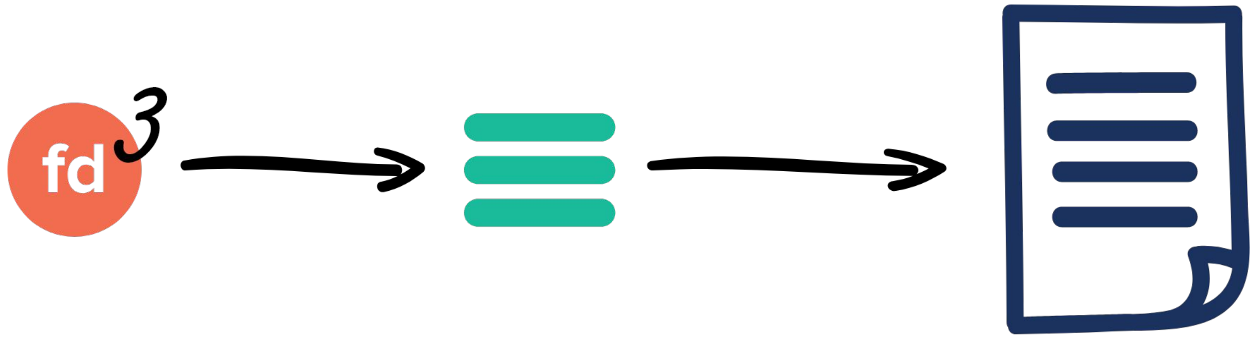

fd
per process
across processes


dups
fd
fd
fd
fd
fd
process 42
process 23
same actual file
inodes, directory entries, pipes, sockets

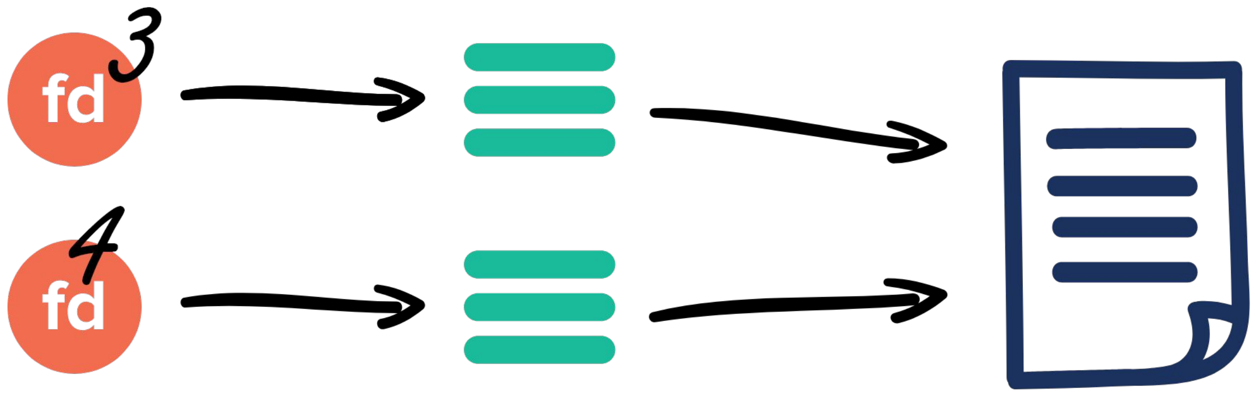
Wait it's all Files
Always has been

The dup System Call

- Duplicates a file descriptor.
- Uses the lowest-numbered available *(for this process)* file descriptor as the new descriptor.
- The old and new descriptors may be used interchangeably, as they point to the same open file description, and share the file offset and the status flags.
- Closing one does not affect the other.

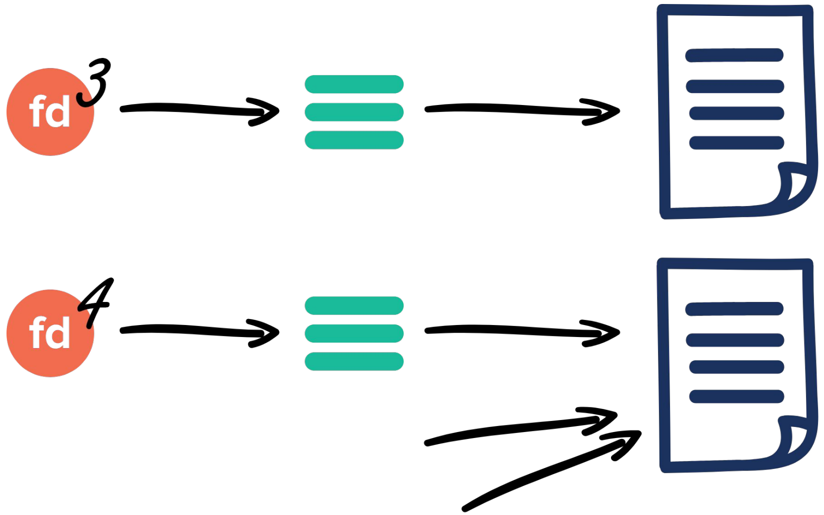


On calling dup

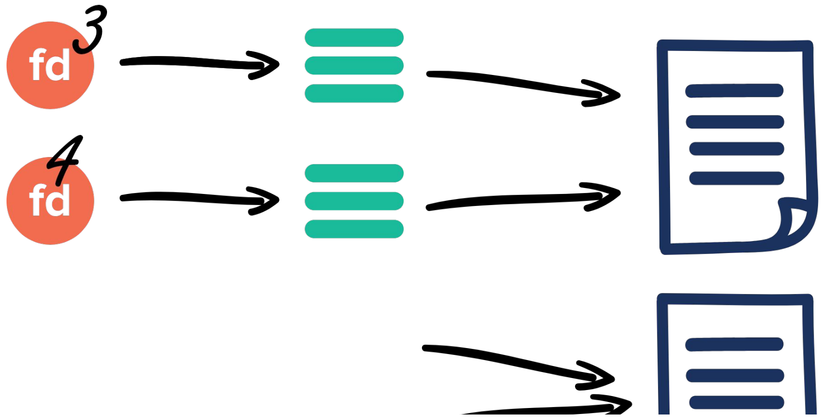


The dup2 System Call

- Same as `dup`, except it uses the specified file descriptor number (`newfd`) as the new file descriptor.
- So, it copies `oldfd` *into* `newfd`.
- If `newfd` was previously open, it closes it before reusing it, *automically*.
- **`dup2(oldfd, newfd)`**



On calling dup2(3, 4)



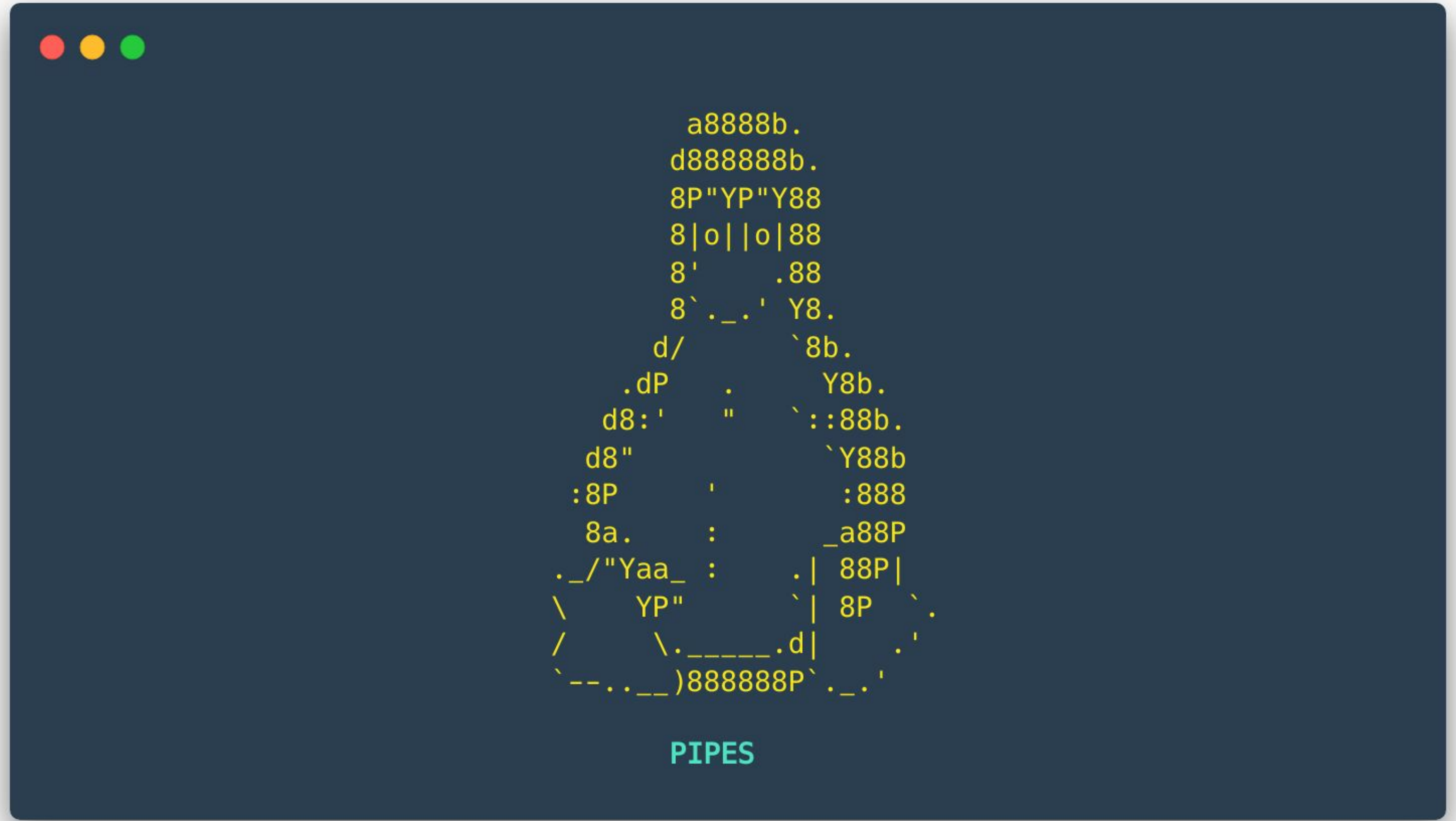
So what would this do?

```c
int fd = open("write-here.txt");
dup2(fd, STDOUT_FILENO);
write(STDOUT_FILENO, "Where would this go?")
```

Function signatures may be simplified, resulting in syntactically invalid C code.

Implementing Input Redirection

```
if input redirection needed:
    backup STDIN_FILENO
    duplicate input file into STDIN_FILENO

execute command

if input redirection was done:
    restore STDIN_FILENO
```



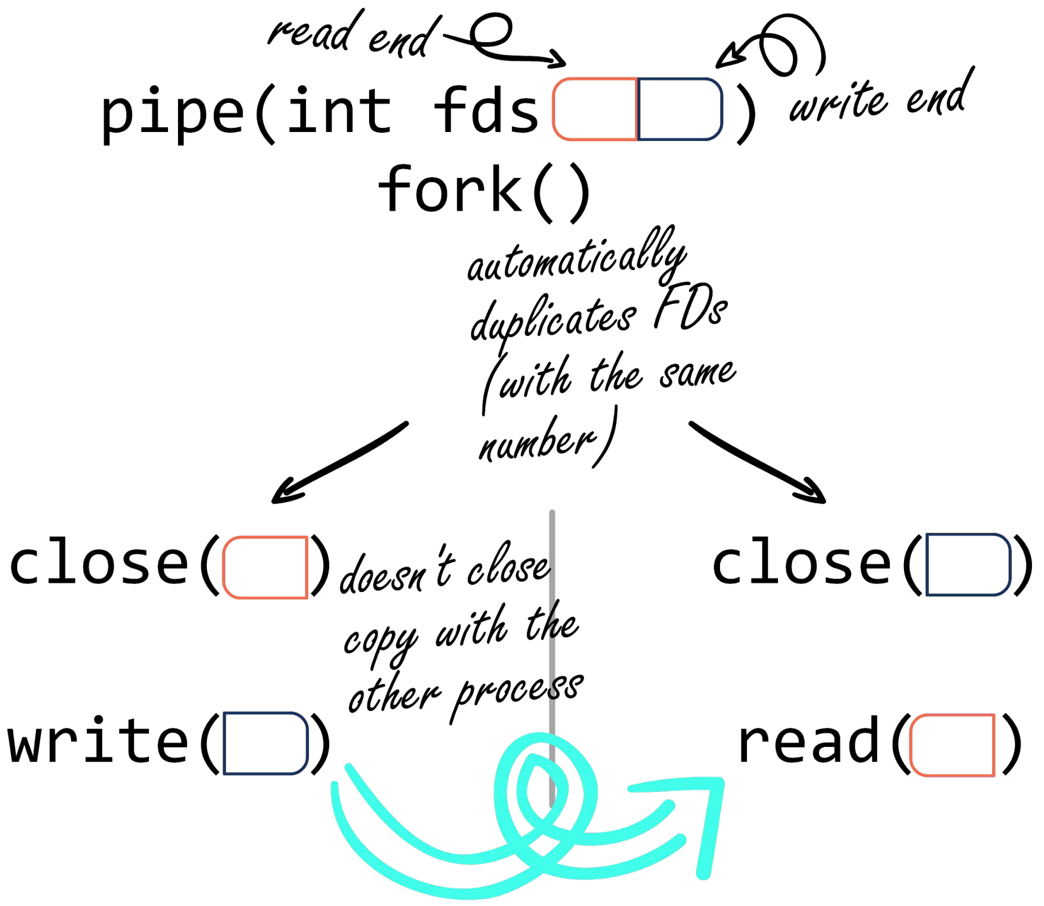
```
        a8888b.
       d888888b.
       8P"YP"Y88
       8|o||o|88
       8'    .88
       8`._.' Y8.
      d/      `8b.
    .dP   .     Y8b.
   d8:'   "   `::88b.
  d8"           `Y88b
 :8P    '       :888
  8a.    :      _a88P
._/"Yaa_ :    .| 88P|
\    YP"      `| 8P  `.
/     \._____.d|    .'
`--..__)888888P`._.'

       PIPES
```

Pipe

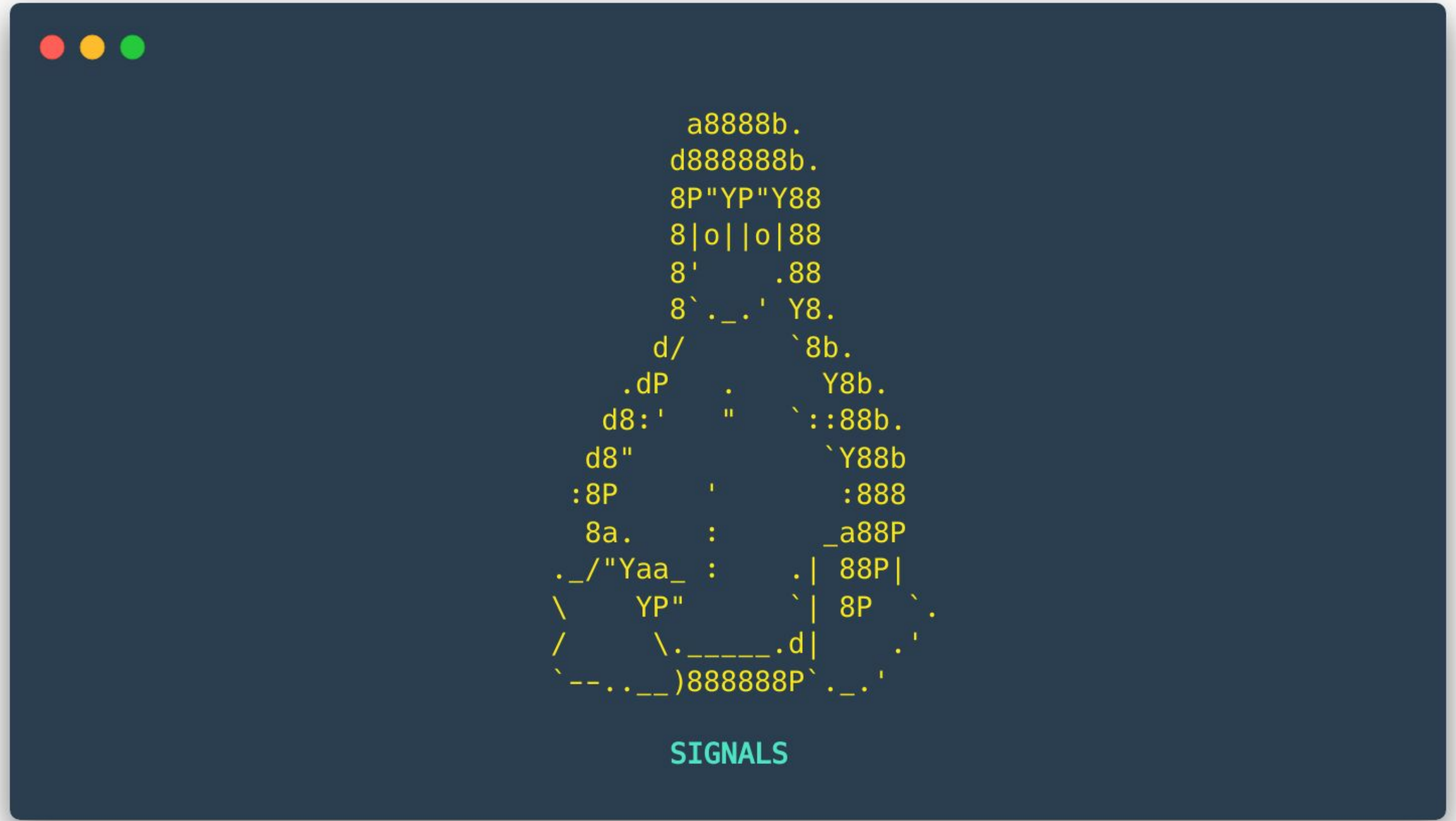
- Conceptually, a connection between two processes, such that one can *write into the pipe*, and the other can *read from it*.
- **`echo say-something.txt | wc`**

The `pipe` System Call

- **Opens a pipe**, an area in **main memory** which is treated as a *virtual file*.
- This pipe can be used by the creating process, as well as its descendants.
- Returns *two* file descriptors, one pointing to the **read-end** of the pipe, and the other to the **write-end**.
- One descendant can write to this file, and the other can read from it.


read end
pipe(int fds 🟧🟦)
write end
fork()
automatically duplicates FDs (with the same number)
close(🟧)
doesn't close copy with the other process
close(🟦)
write(🟦)
read(🟧)

Function signatures may be simplified, resulting in syntactically invalid C code.

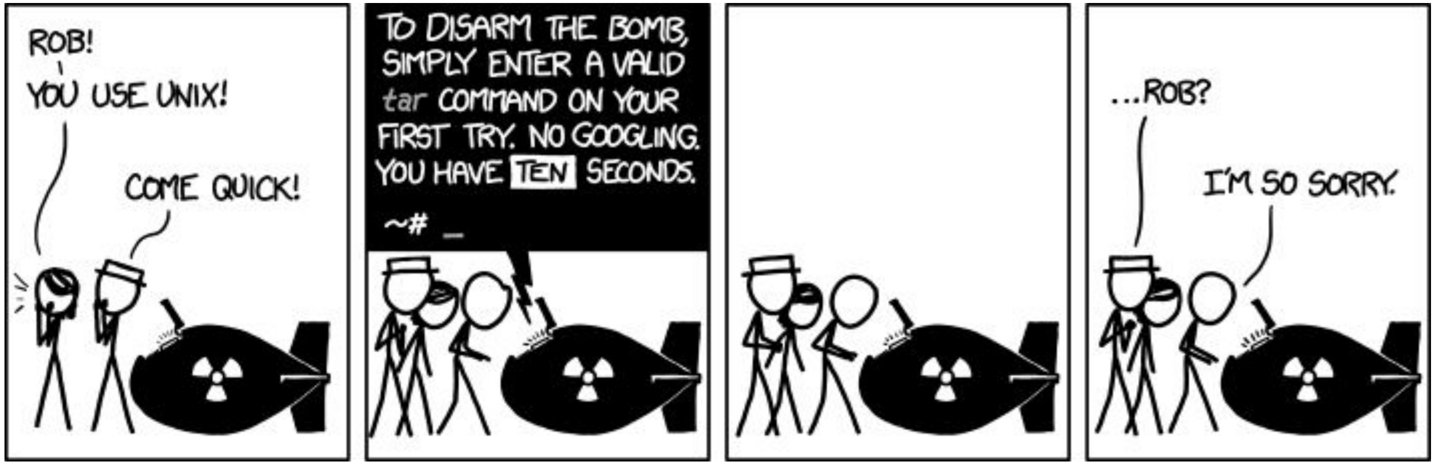


```
        a8888b.
       d888888b.
       8P"YP"Y88
       8|o||o|88
       8'    .88
       8`._.' Y8.
      d/      `8b.
    .dP   .     Y8b.
   d8:'   "   `::88b.
  d8"           `Y88b
 :8P     '       :888
  8a.    :      _a88P
._/"Yaa_ :    .| 88P|
\    YP"      `| 8P  `.
/     \._____.d|    .'
`--..__)888888P`._.'
```

SIGNALS

The `signal` System Call

- Every signal has a *default* **disposition**, some terminate the process, some are ignored, some stop the process, etc.
- Using `signal`, a process can *elect* one of the following behaviours:
  - perform the default action
  - ignore the signal
  - catch the signal with a custom signal handler


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

Source: xkcd.com

Please compress your submissions correctly.