**Pipe example**

**Goal:** (functionality of `ls | wc`)

```
+-----------------+       +-----------------+    +-----------------+
|  tty            | ->    |  sh             | -> |  tty            |
| (terminal)      |       | 0               | -> | 1               |    | 0               |
```

**Key**
- **pid:** process with command `cmd` and process id `pid` (parent/child relation)
- **pipe:** pipe: left is write, right is read
- **i:** file descriptor i, open for reading
- **j:** file descriptor j, open for writing

**Initially:**
```
+-----------------+       +-----------------+    +-----------------+
|  tty            | ->    |  sh             | -> |  tty            |
```

**Steps**
1) `sh (1001)` executes `pipe(fd)`

```
+-----------------+       +-----------------+    +-----------------+
|  tty            | ->    |  sh             | -> |  tty            |
```

**Creates a pipe and two new open file descriptors:**
- `fd[0]`: read
- `fd[1]`: write
2) sh (1001) executes fork()

3) sh(1001) executes close(fd[1])
sh(1017) executes
  close(fd[0]), dup2(fd[1],1),
  close(fd[1]), exec(ls)

creates a new process: sh (1017) which has
copies of its parent's open file
descriptors
0,1, fd[0], fd[1]

closes unneeded
file descriptors,
replaces process
1017 fd 1
by write fd
for pipe,
starts ls
4) `sh (1001)` executes `fork()` creates a new process: `sh (1027)` which has copies of its parent's open file descriptors: 0, 1, `fd[0]`

5) `sh (1001)` executes `close(fd[0])` and waits for both child processes `(1017, 1027)` to terminate. `sh (1027)` executes `dup2(fd[0], 0), close(fd[0]), exec(wc)` closes unneeded file descriptor replaces process `1027` `fd 0` by `read fd` for pipe starts `wc`

( `sh (1001)` suspends execution until the two child processes terminate )

[Goal achieved ]
Another view of this is how information in the process table changes.

Initially:

<table>
<thead>
<tr>
<th>pid=1001</th>
<th>open fds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tty, r</td>
</tr>
<tr>
<td>1</td>
<td>tty, w</td>
</tr>
<tr>
<td>2</td>
<td>tty, w</td>
</tr>
</tbody>
</table>

1) sh (1001) executes pipe(fd)
(assuming it returns fd[0] = 3 and fd[1] = 4.)

Then:

- new fds created for pipe

<table>
<thead>
<tr>
<th>pid=1001</th>
<th>open fds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tty, r</td>
</tr>
<tr>
<td>1</td>
<td>tty, w</td>
</tr>
<tr>
<td>2</td>
<td>tty, w</td>
</tr>
<tr>
<td>3</td>
<td>pipe, r</td>
</tr>
<tr>
<td>4</td>
<td>pipe, w</td>
</tr>
</tbody>
</table>

2) sh (1001) executes fork()

Then:

- open fds copied to child process

<table>
<thead>
<tr>
<th>pid=1017</th>
<th>open fds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tty, r</td>
</tr>
<tr>
<td>1</td>
<td>tty, w</td>
</tr>
<tr>
<td>2</td>
<td>tty, w</td>
</tr>
<tr>
<td>3</td>
<td>pipe, r</td>
</tr>
<tr>
<td>4</td>
<td>pipe, w</td>
</tr>
</tbody>
</table>
3) Suppose sh(1001) has executed close(fd[1]) and sh(1017) has executed close(fd[3]).

Then:

- Entry for 4 removed from 1001's table.

Entry for 3 removed from 1017's table.

3 continued) Suppose sh(1017) has now executed dup2(fd[1], 1).

Then:

- Entry in 1017's table for 4 copied to 1.

Table entries:

- **pid=1001**
  - open fds:
    - 0: tty, r
    - 1: tty, w
    - 2: tty, w
    - 3: pipe, r

- **pid=1017**
  - open fds:
    - 0: tty, r
    - 1: pipe, w
    - 2: tty, w
    - 3: pipe, w

[And so on.]