Advanced Programming in the UNIX Environment

Week 09, Segment 1:
socketpair(2)

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

pipe(2)

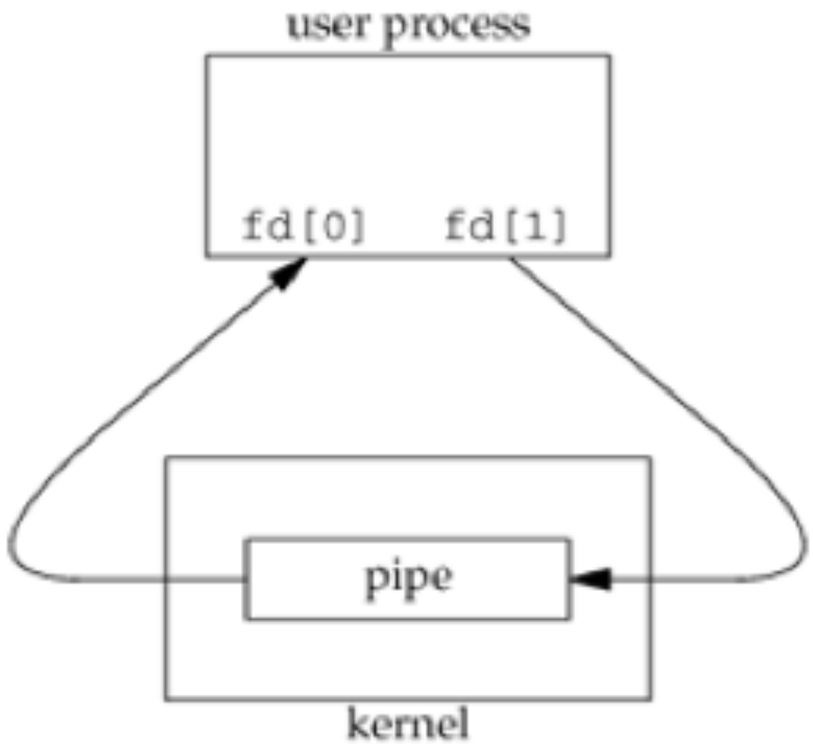
```
#include <unistd.h>
int pipe(int filedes[2]);
```

Returns: 0 ok, -1 otherwise

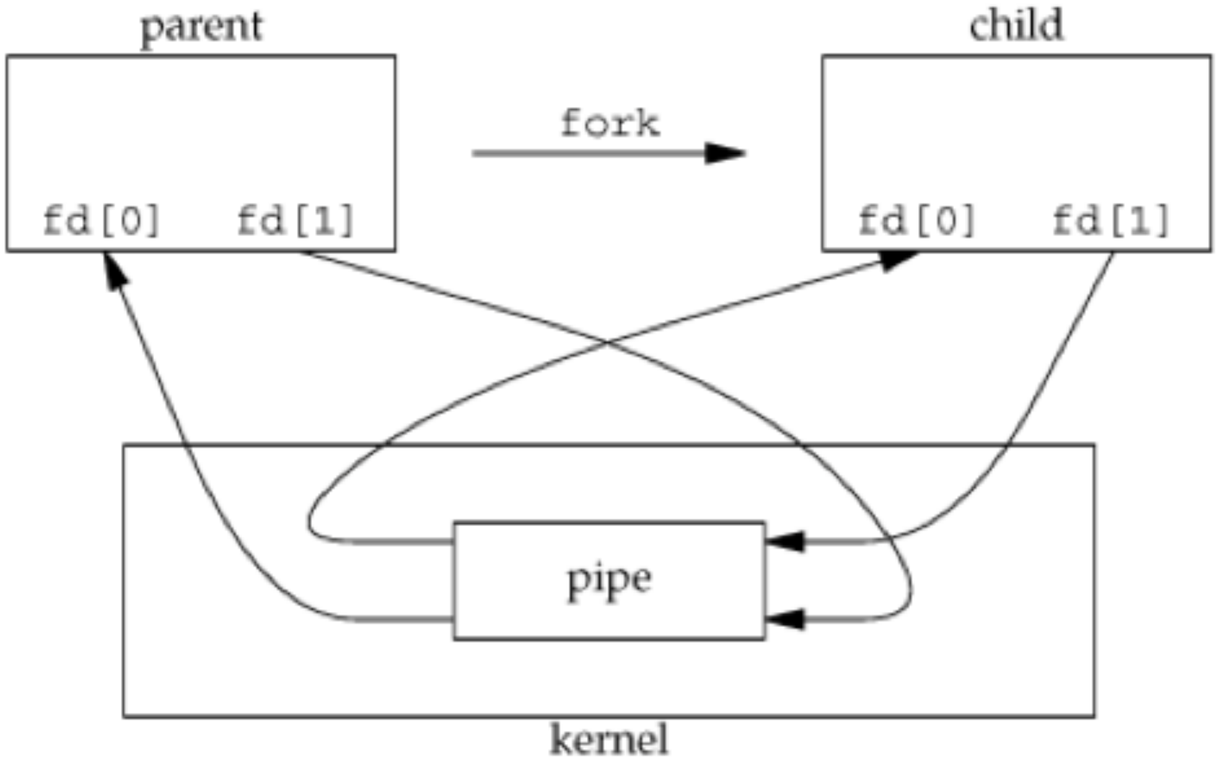
```
$ proc1 | proc2
```

- oldest and most common form of IPC
- usually unidirectional / half-duplex

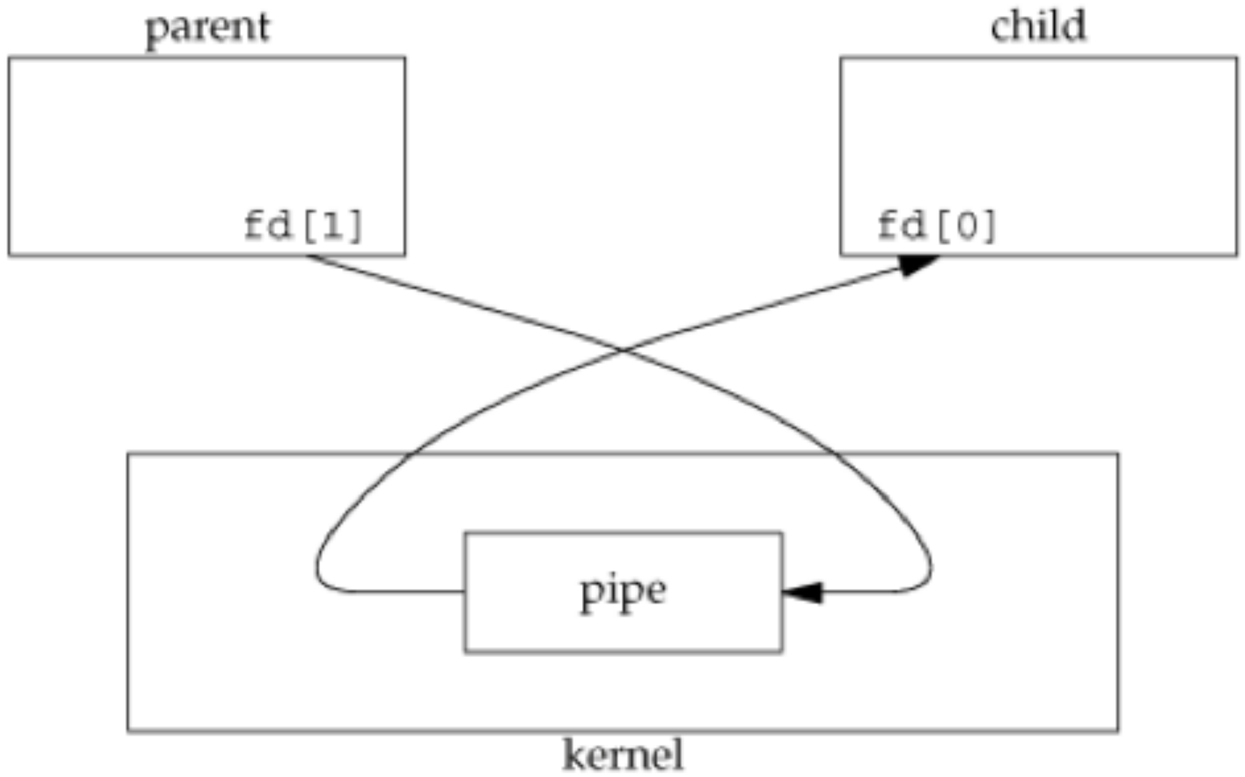
pipe(2)



pipe(2)



pipe(2)



pipe(2)

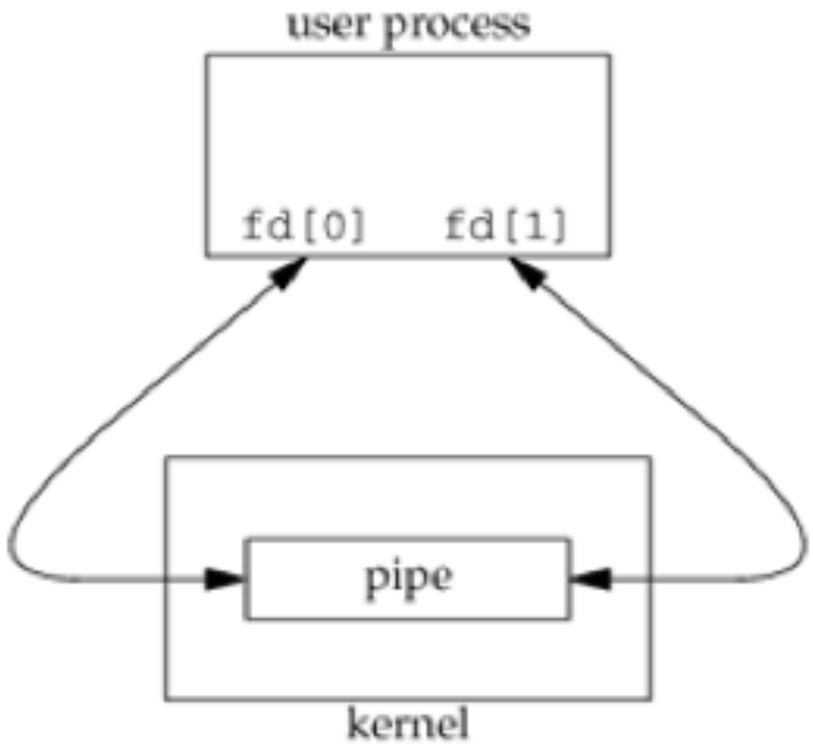
```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol, int *sv);
```

Returns: 0 ok, -1 otherwise

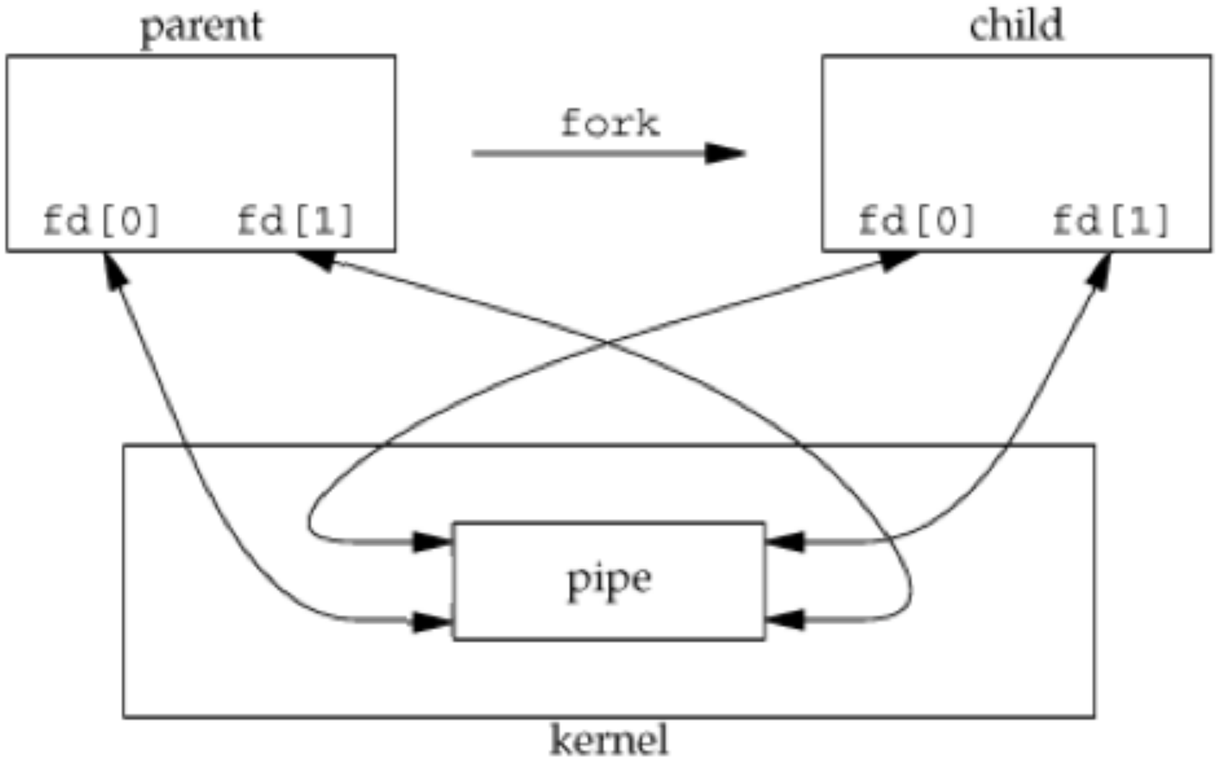
The socketpair(2) call creates an unnamed pair of connected sockets in the specified *domain*, of the specified *type*, and using the optionally specified *protocol*.

The descriptors used in referencing the new sockets are returned in sv[0] and sv[1]. The two sockets are indistinguishable.

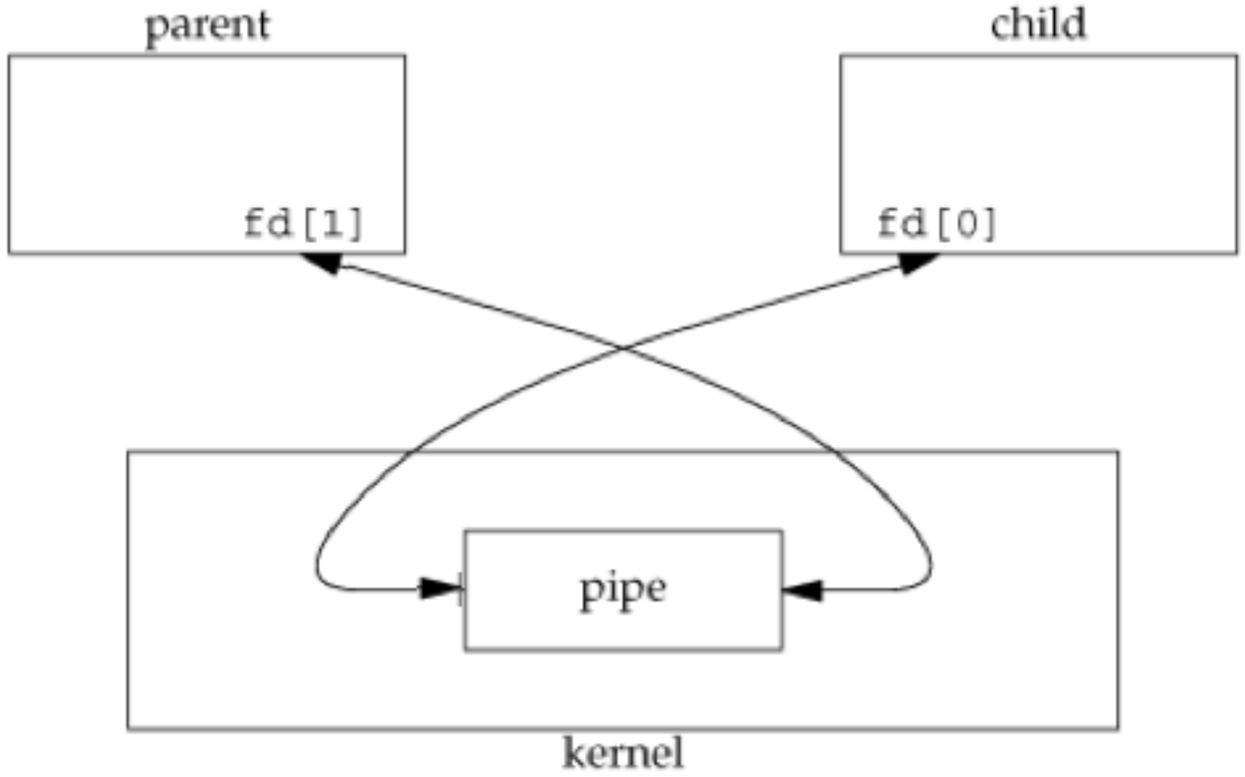
socketpair(2)



socketpair(2)



socketpair(2)



Terminal — 80×24

```
jschauma@apue$ vim socketpair.c
jschauma@apue$ cc -Wall -Wextra -Werror socketpair.c
jschauma@apue$ ./a.out
Parent (1421) --> sending: In Xanadu, did Kubla Khan . . .
Child  (1948) --> sending: A stately pleasure dome decree . . .
Parent (1421) --> reading: A stately pleasure dome decree . . .
Child  (1948) --> reading: In Xanadu, did Kubla Khan . . .
jschauma@apue$
```

Questions

- What happens if you change the *type* and/or *protocol* in the socketpair(2) call?
- Can you change `socketpair.c` to read/write from both ends?
- What happens if you change the order of the `read(2)` and `write(2)` calls in both the parent and the child?

- See also: /usr/share/doc/reference/ref3/sockets/sockets.txt