

UNIX Domain Sockets

W4118 Operating Systems I

<https://cs4118.github.io/www/2024-1/>

Shared Memory

1. Multiple threads in a single process
 - Process address space is already shared among the threads
2. **Related** processes (i.e., parent and child)
 - Anonymous mmap
3. **Unrelated** processes
 - File-backed mmap

Synchronization

1. Multiple threads in a single process
 - pthread mutex, condition variable,...
2. Multiple processes **with some shared memory**
 - pthread mutex, condition variable,...
 - Unnamed POSIX semaphore
3. Multiple process **with no shared memory**
 - Named POSIX semaphore

Data Passing

1. **Related** processes

- Unnamed pipe
 - i. Created by `pipe()`
 - ii. half-duplex

2. **Unrelated** processes

- Named pipe (aka FIFO)
 - i. Created by `mkfifo()`
 - ii. half-duplex

3. **Distant** processes

- TCP/UDP sockets
- full duplex
- reliable/unreliable and high/low overhead

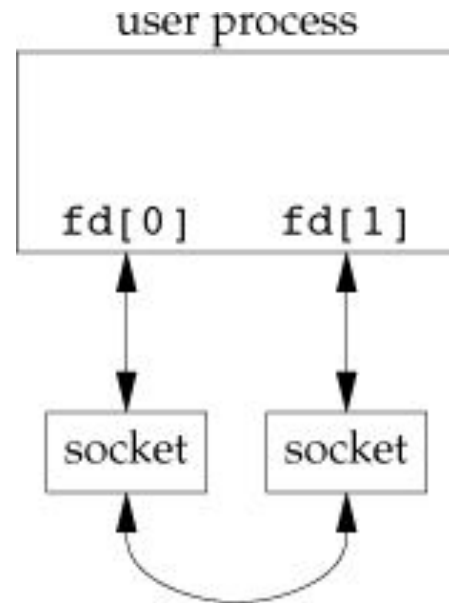
Best of both worlds: UNIX domain sockets

1. Unnamed pair of connected sockets for related processes
 - a. Created by `socketpair(AF_UNIX, ...)`
 - b. Just like a pipe but full duplex
2. Named local-only socket for unrelated processes
 - a. created by `socket(AF_UNIX, ...)`
 - b. represented by a special file
3. Reliable when used in datagram mode
4. Can transport special things like an open file descriptor

UNIX domain sockets

```
int socketpair(int domain, int type, int protocol, int sv[2]);  
  
// Returns 0 if OK, -1 on error
```

Same picture as the one for `pipe()` but
arrows going both ways



Passing open file descriptors

