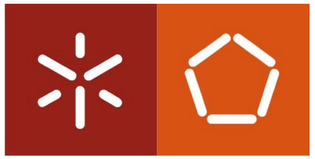
Operating Systems

(Sistemas Operativos)

Guide 5: FIFOs

University of Minho

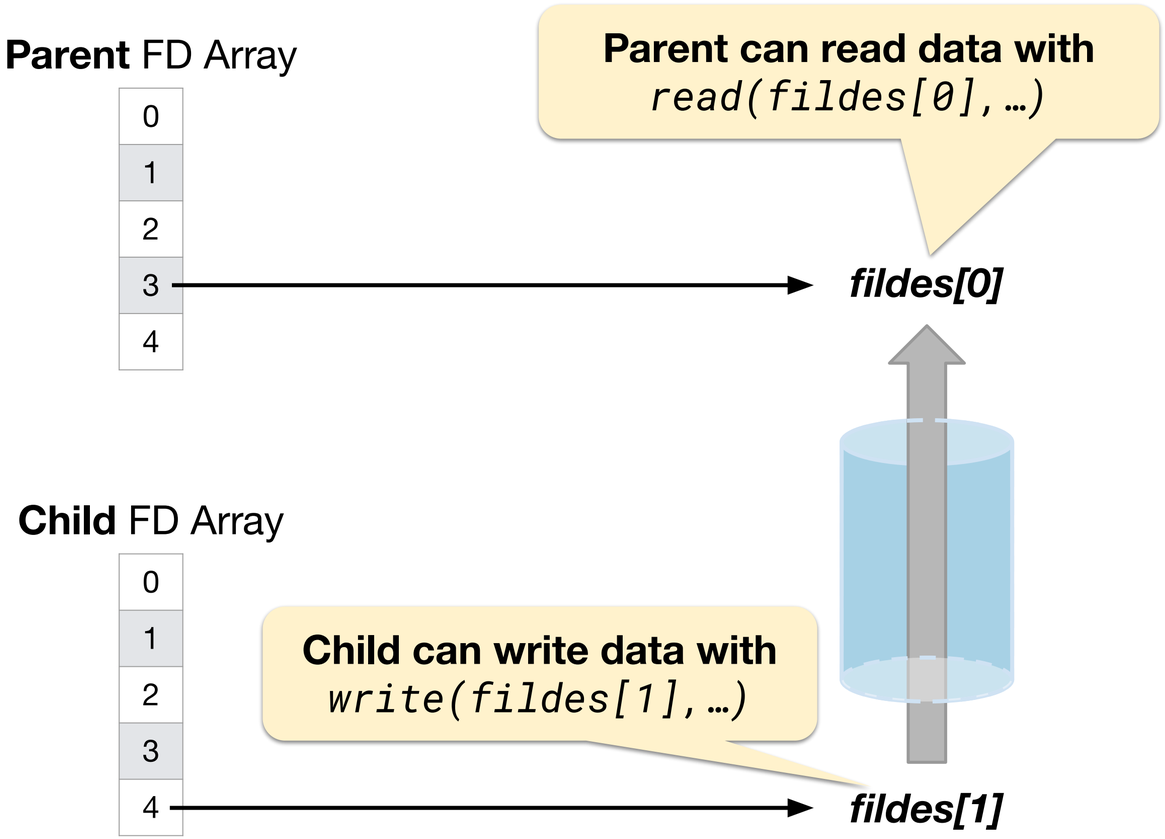
2024 - 2025



Process API

Anonymous Pipes (recap)

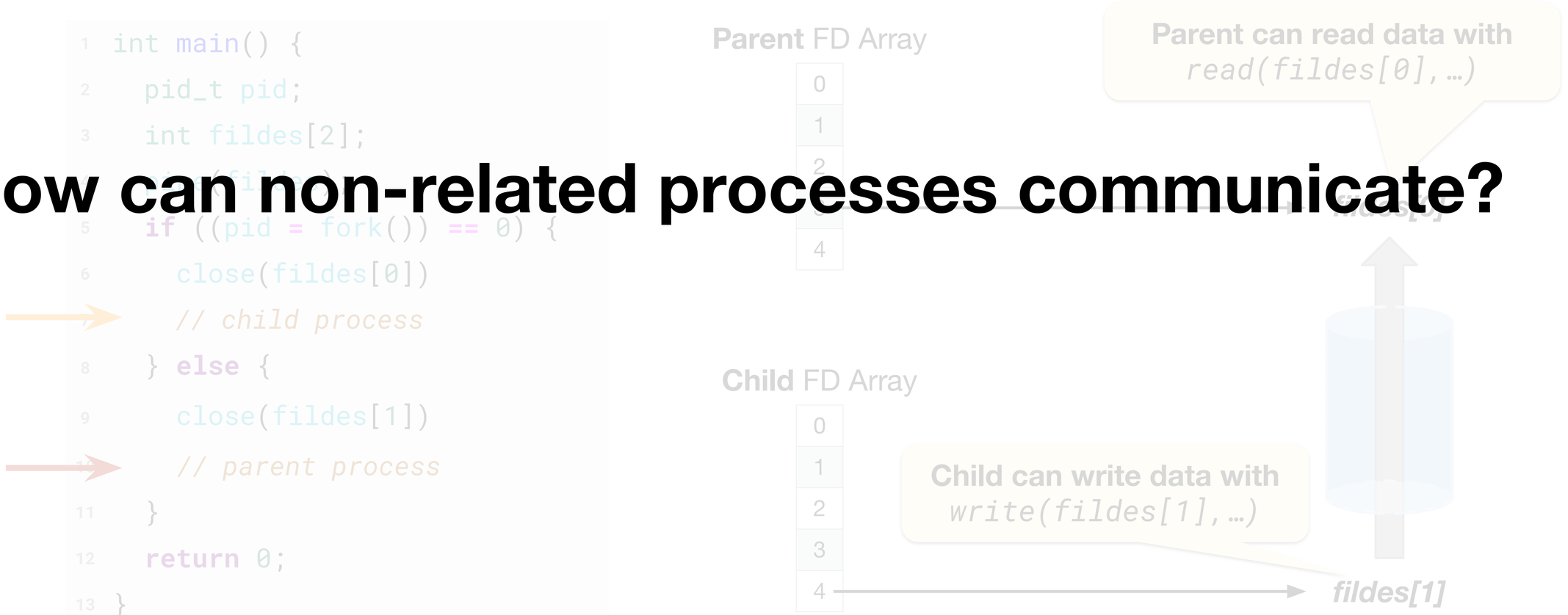
```c
int main() {
  pid_t pid;
  int fildes[2];
  pipe(fildes);
  if ((pid = fork()) == 0) {
    close(fildes[0])
    // child process
  } else {
    close(fildes[1])
    // parent process
  }
  return 0;
}
```



Process API

Anonymous Pipes (recap)

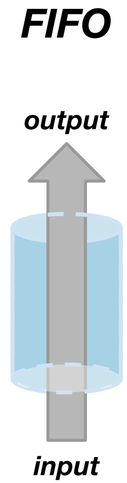
```c
int main() {
  pid_t pid;
  int fildes[2];
  pipe(fildes);
  if ((pid = fork()) == 0) {
    close(fildes[0])
    // child process
  } else {
    close(fildes[1])
    // parent process
  }
  return 0;
}
```



How can non-related processes communicate?

Process API

Inter-Process Communication



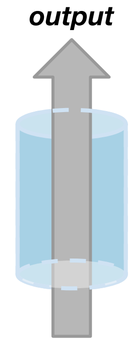
Named Pipes

- **Communication across non-related processes**
- Produced (written) data is **kept in a memory region** to be consumed (read).
- The kernel handles writers (producers) and readers (consumers).
- **Writers block (wait) if there is no available space, and readers block if there is no data.**
- Data flows in a **one-way First-in First-out** (**FIFO**) manner.

Process API

Named Pipes

pathname



#include <sys/types.h>
#include <sys/stat.h>

- *int* ***mkfifo****(const char path** **pathname**, *mode_t* **mode***)*
  - **pathname:** absolute or relative pathname to the special FIFO file
  - **mode**: file permissions
    - 0600 - owner of the file can read/write
  - Returns: 0 on success, -1 otherwise

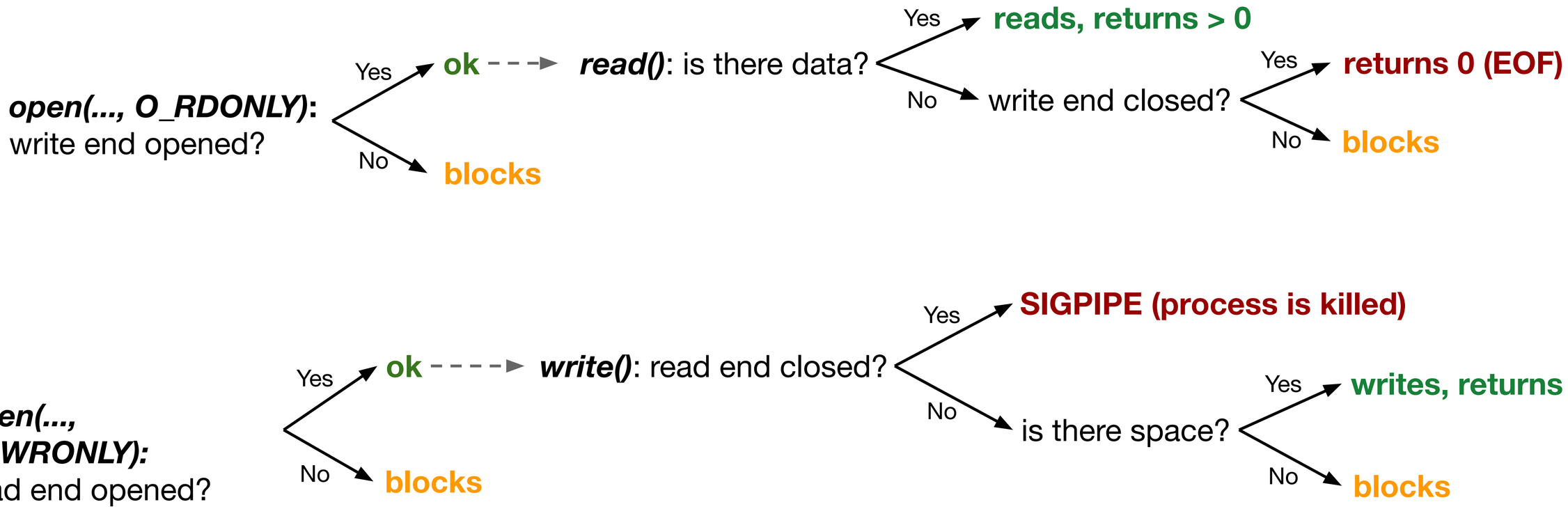
For more information: *$ man 3 mkfifo*

Considerations:

1. ***mkfifo* creates the FIFO special file**
2. **Accessed through syscalls** (*open*, *close*, *read*, *write*)
   a. **Explicitly removed with** *unlink*
   b. **Cannot** change file pointer (*lseek*)
3. **Unidirectional communication**
4. **Reads and writes block**

Process API

Named Pipes



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
...
```

Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```

Created FIFO
Check it: *$ ls -l*
(How do you know it is a pipe?)

Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

no write end opened blocks…

FD Array

0
1
2
3

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```

Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

write end is opened
continue

read end is opened
continue

FD Array

0
1
2
3

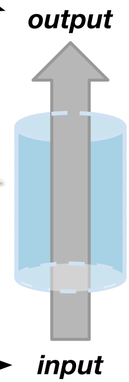
FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```

FIFO in main memory



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RD
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

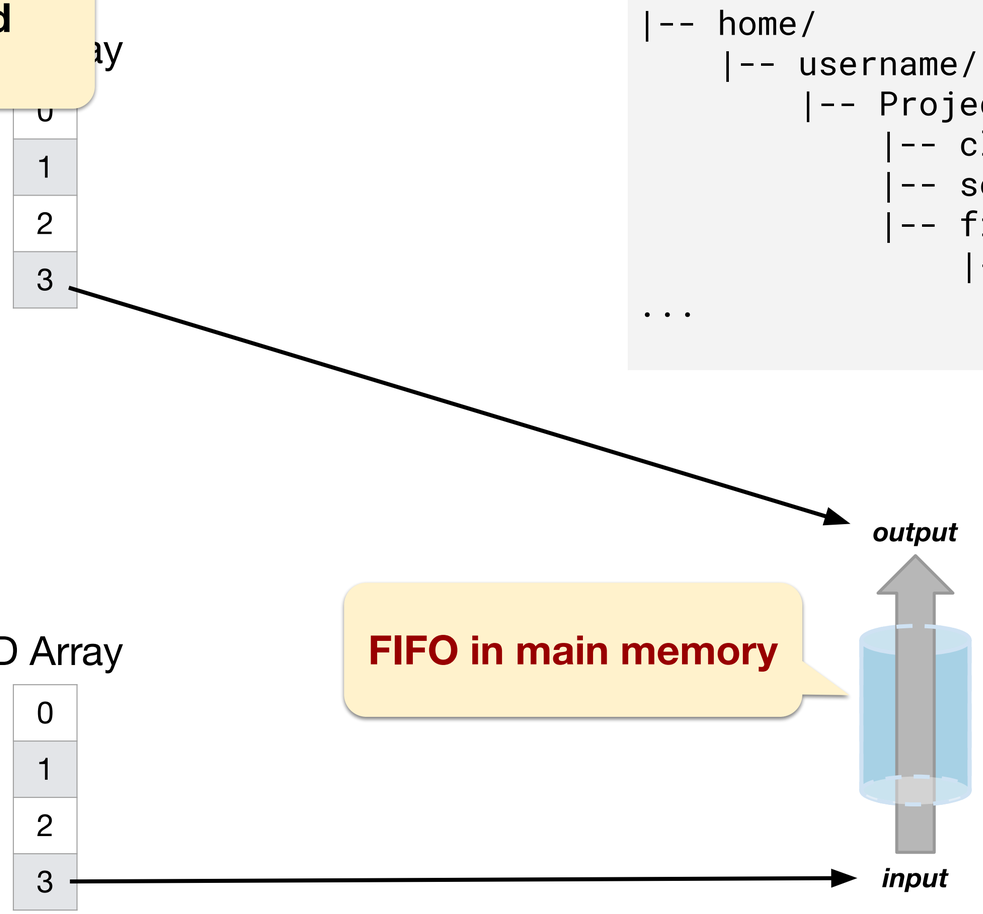
PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

nothing to read blocks...

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RD
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

reads data from FIFO

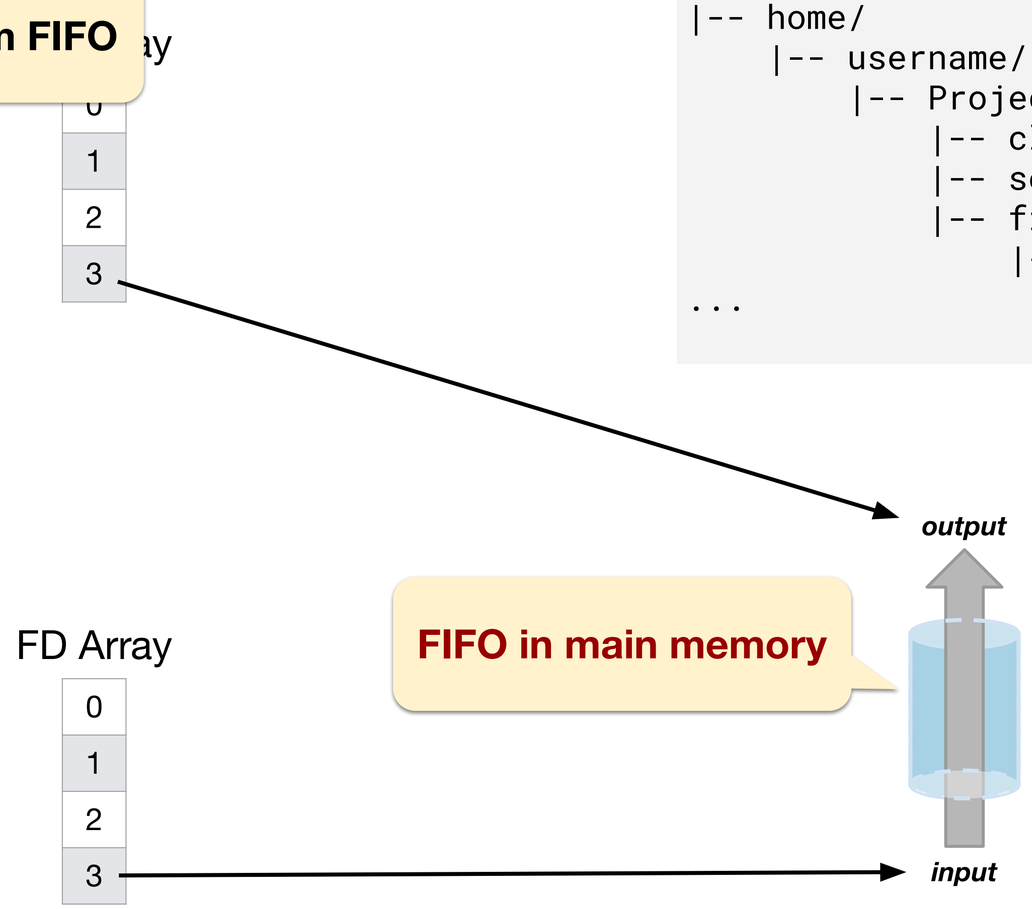
PID 5391 (

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

writes data to FIFO

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```



Process API

Example: communicating via FIFO

PID 2034 (server)

```
1  int main() {
2    mkfifo("fifos/myfifo", 0666);
3    int fifo = open("fifos/myfifo", O_RD
4    char buf[BUFSIZE];
5    int n;
6    while((n = read(fifo, buf, BUFSIZE)) > 0) {
7      ...
8    }
9    close(fifo);
10   unlink(fifo);
11 }
```

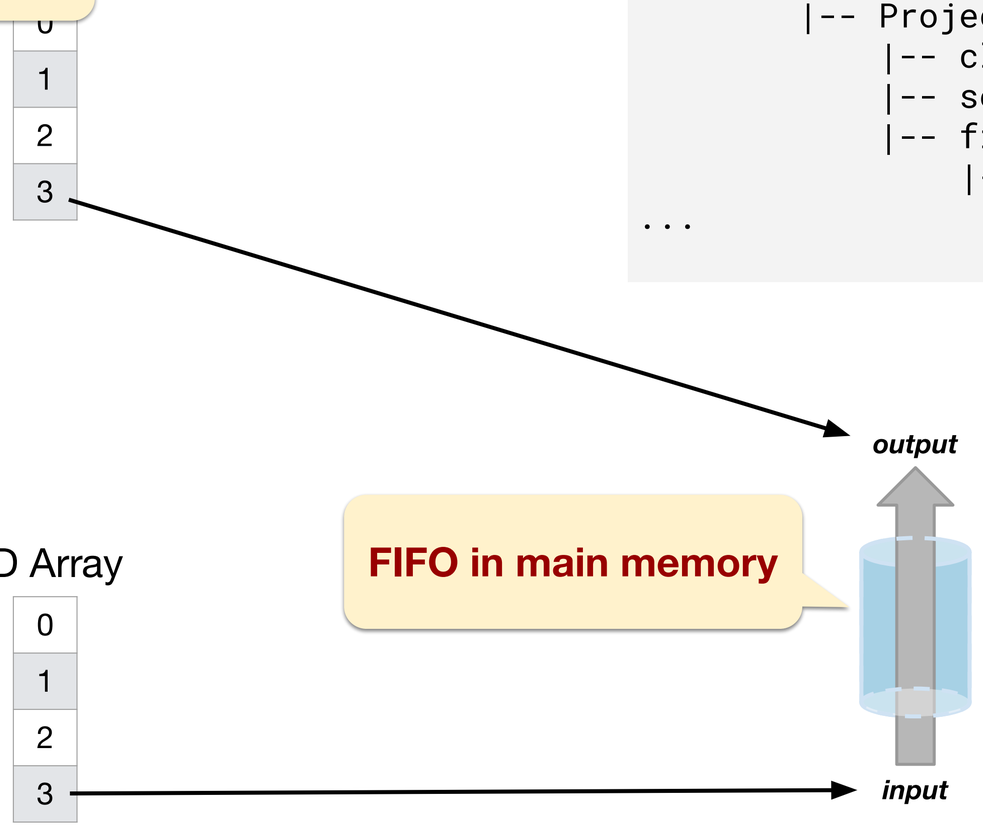
PID 5391 (client)

```
1  int main() {
2    int fifo = open("fifos/myfifo", O_WRONLY);
3    ...
4    write(fifo, buf, BUFSIZE);
5    close(fifo);
6  }
```

blocks again

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RD
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

unblocks and read returns 0 (EOF)

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/my
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

closes FIFO (without deleting it)

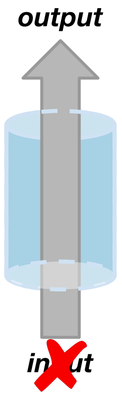
FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```

FIFO in main memory



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, bu
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

closes FIFO (without deleting it)

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

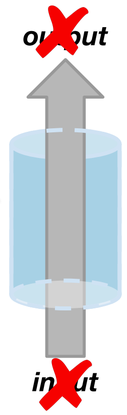
FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```

FIFO in main memory



Process API

Example: communicating via FIFO

PID 2034 (server)

```c
int main() {
  mkfifo("fifos/myfifo", 0666);
  int fifo = open("fifos/myfifo", O_RDONLY);
  char buf[BUFSIZE];
  int n;
  while((n = read(fifo, buf, BUFSIZE)) > 0) {
    ...
  }
  close(fifo);
  unlink(fifo);
}
```

Deletes FIFO after all processes close it

PID 5391 (client)

```c
int main() {
  int fifo = open("fifos/myfifo", O_WRONLY);
  ...
  write(fifo, buf, BUFSIZE);
  close(fifo);
}
```

FD Array

0
1
2
3

Filesystem

```
/
|-- home/
    |-- username/
        |-- Project
            |-- client
            |-- server
            |-- fifos/
                |-- myfifo
...
```