# Interprocess communication using unnamed channels (pipes)

## What we use

**#include <unistd.h>**

## What we have

|  |  |
| --- | --- |
| int filedes[2]; | // The array pipefd is used to return two file descriptors referring to the ends of the pipe  // pipefd[0] refers to the read end of the pipe.  // pipefd[1] refers to the write end of the pipe. |
| int pipe(int filedes[2]); | // creating pipe using filedes  // 0 if successful, –1 if successful. |
| close(file\_desr) |  |
| write(FD, char\* msg, N) |  |
| read(FD, char\* msg, K) |  |
| and all function from HW #1 | |

## Theoretical bases

Inter-process communication is carried out in the following ways:

• Environment variables

• Signals

• Channels (**pipes**)

◦ Named

◦ **Unnamed**

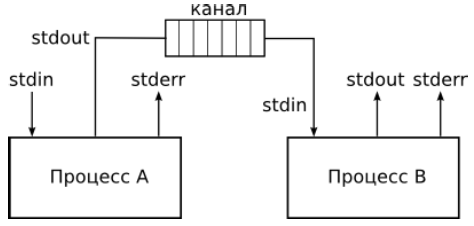
• Sockets

Channel (pipe) is a data stream between two or more processes that has an interface similar to reading or writing to a file.

Channels have two limitations:

1. Historically, they are **simplex** (that is, data can be transmitted over them in only one direction);

2. Channels can only be used to organize interaction between processes that have a common ancestor. Typically, a channel is created by the parent process, which then calls the fork () function, after which this channel can be used to communicate between the parent and child processes.



## Task for homework

1. Parallel computing pi-value

Pi=4 − 4/3 + 4/5 − 4/7 + … + ( (-1)^(n+1)\*4)/(2\*n-1)

n -- number of terms

k -- number of processes

1. **Draw graphs dependency: OY is time, and OX is number of processes**

For drawing graphs you can use MathCad, Matlab, Python, Octave, R and others math tools

Be careful with edges for intervals for pi calculations

## References

<http://man7.org/linux/man-pages/man3/pipe.3p.html>