Linux Internals & Networking

System programming using Kernel interfaces

Team Emertxe

Contents

Linux Internals & Networking
Contents

EMERTXE

Inter Process Communications (IPC)

Communication
In real world

- Face to face
- Fixed phone
- Mobile phone
- Skype
- SMS

Inter Process Communications
Introduction

- *Inter process communication (IPC)* is the mechanism whereby one process can communicate, that is exchange data with another processes
- There are two flavors of IPC exist: System V and POSIX
- Former is derivative of UNIX family, later is when standardization across various OS (Linux, BSD etc..) came into picture
- Some are due to “UNIX war” reasons also
- In the implementation levels there are some differences between the two, larger extent remains the same
- Helps in portability as well

Inter Process Communications
Introduction

- IPC can be categorized broadly into two areas:

Data exchange

Communication

- Pipes
- FIFO
- Shared memory
- Signals
- Sockets

Resource usage/access/control

Synchronization

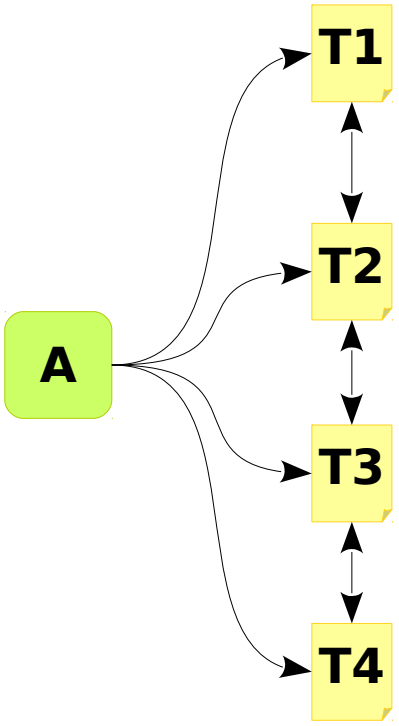
- Semaphores

- Even in case of Synchronization also two processes are talking.

Each IPC mechanism offers some advantages & disadvantages. Depending on the program design, appropriate mechanism needs to be chosen.

Application & Tasks

Example: Read from a file
$ cat file.txt

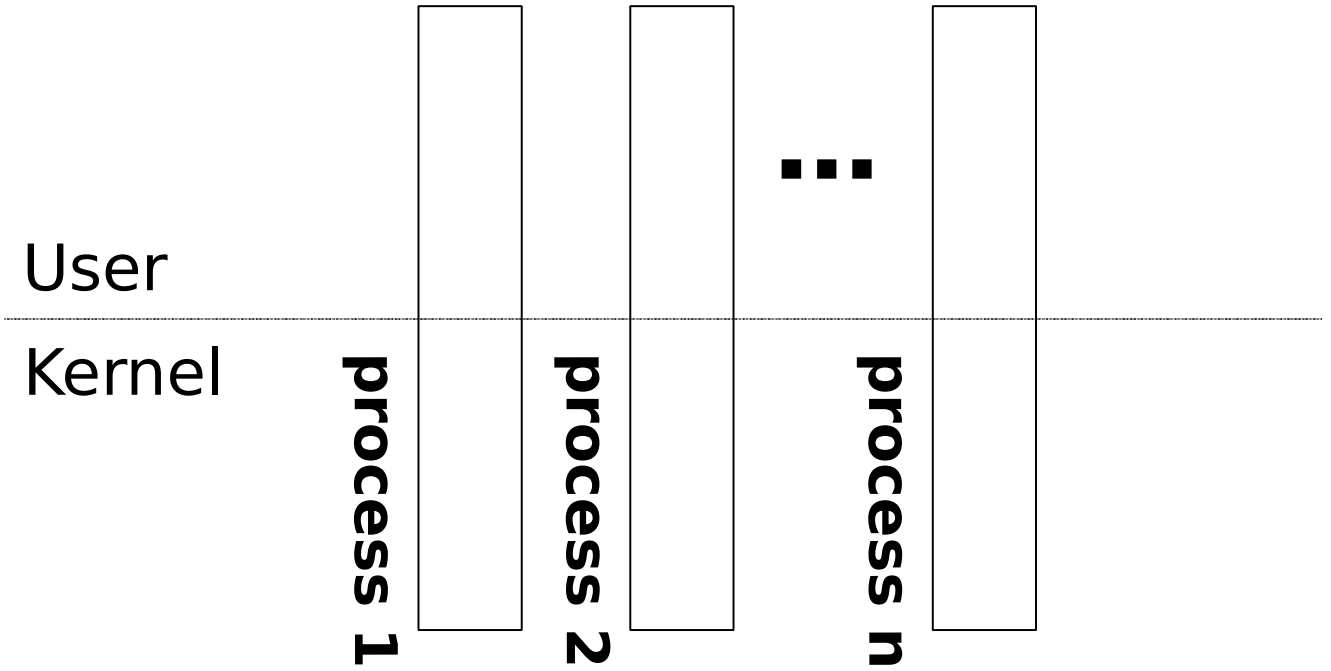


Example: Paper jam handling
in printer

EMERTXE

Inter Process Communications
User vs Kernel Space

- Protection domains - (virtual address space)



How can processes communicate with each other and the kernel? The answer is nothing but IPC mechanisms

Inter Process Communications
Pipes

- A pipe is a communication device that permits unidirectional communication
- Data written to the “write end” of the pipe is read back from the “read end”
- Pipes are serial devices; the data is always read from the pipe in the same order it was written

Inter Process Communications
Pipes - Creation

- To create a pipe, invoke the pipe system call
- Supply an integer array of size 2
- The call to pipe stores the reading file descriptor in array position 0
- Writing file descriptor in position 1

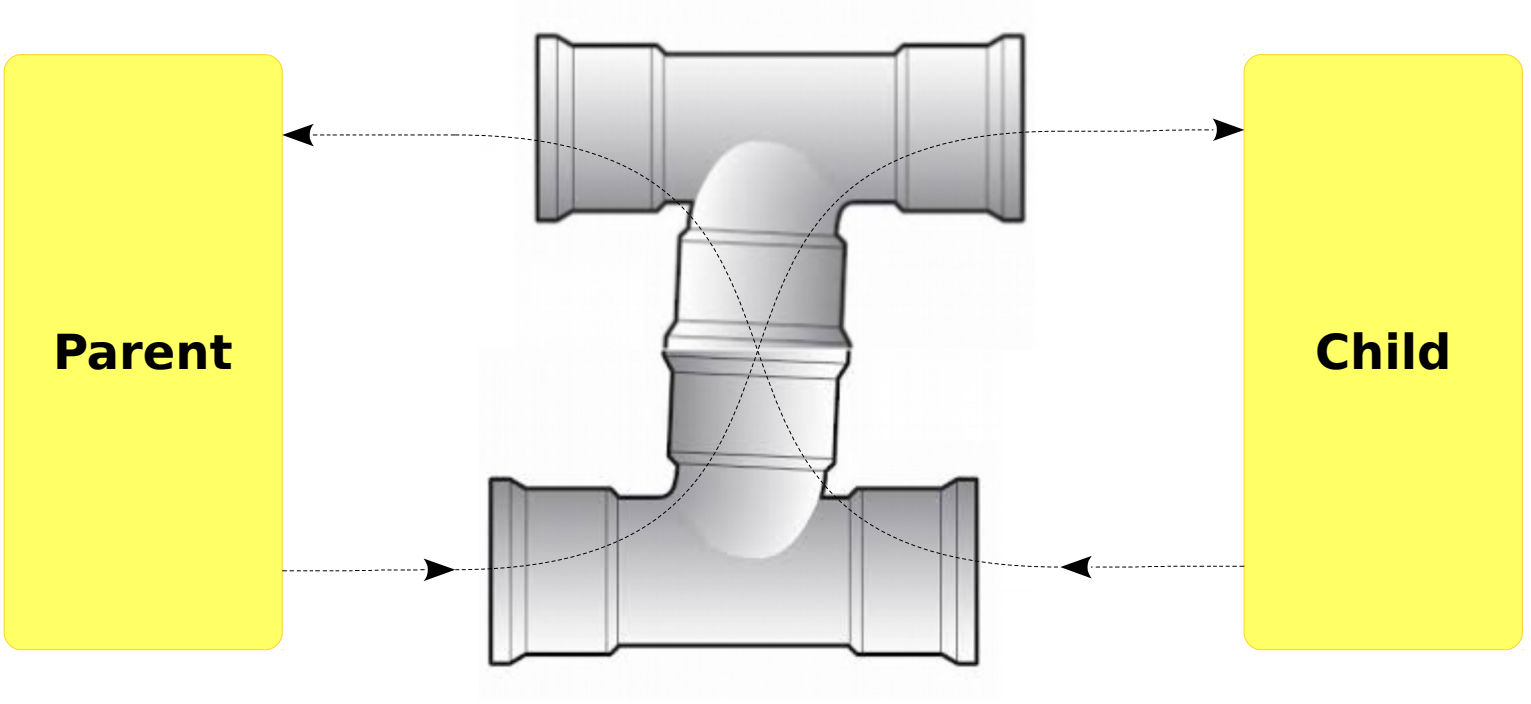
Function	Meaning
int pipe(int pipe_fd[2])	✓ Pipe gets created ✓ READ and WRITE pipe descriptors are populated ✓ RETURN: Success (0)/Failure (Non-zero)

Pipe read and write can be done simultaneously between two processes by creating a child process using fork() system call.

EMERTXE

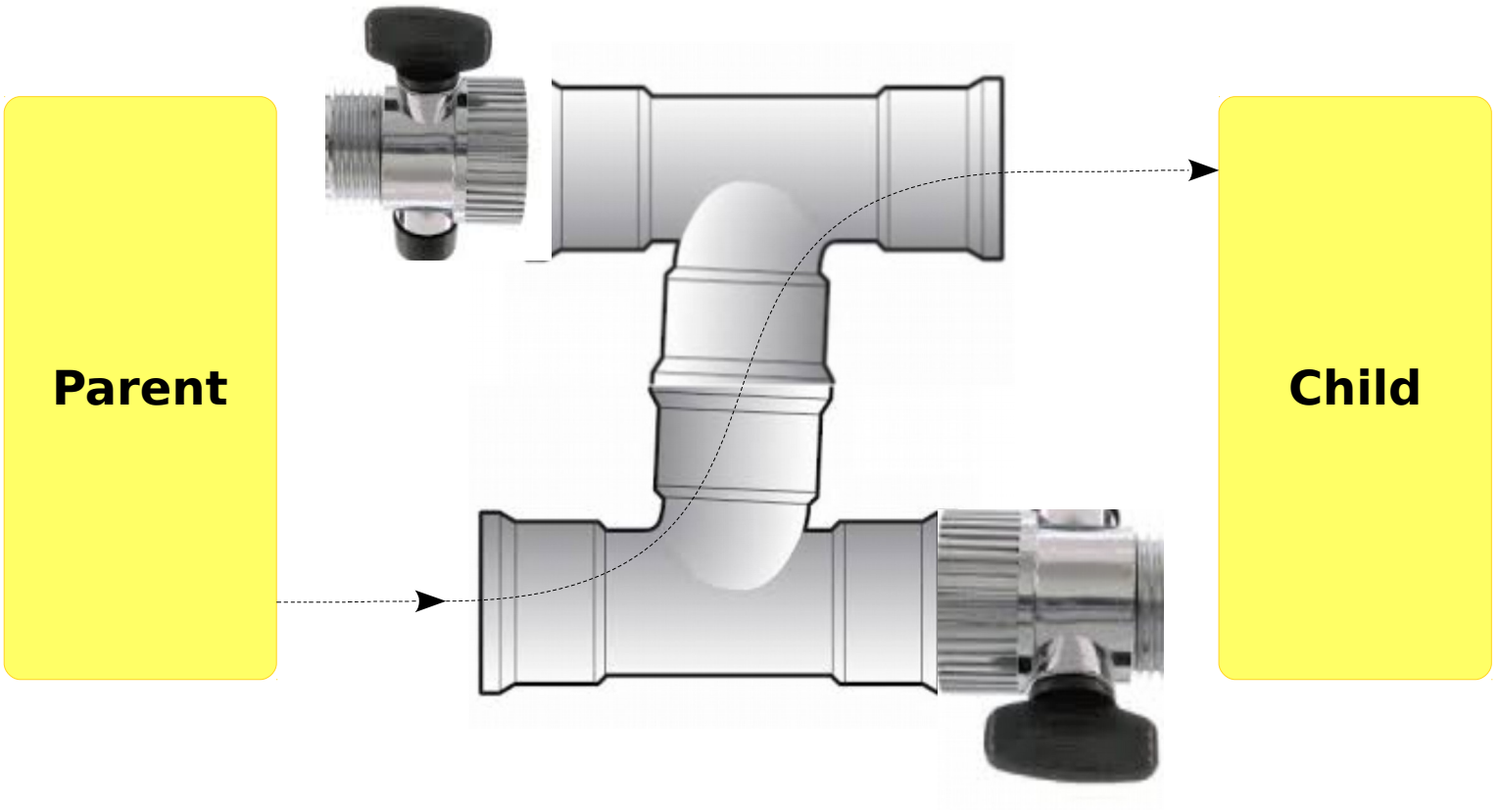
Inter Process Communications
Pipes – Direction of communication

- Let's say a Parent wants to communicate with a Child
- Generally the communication is possible both the way!

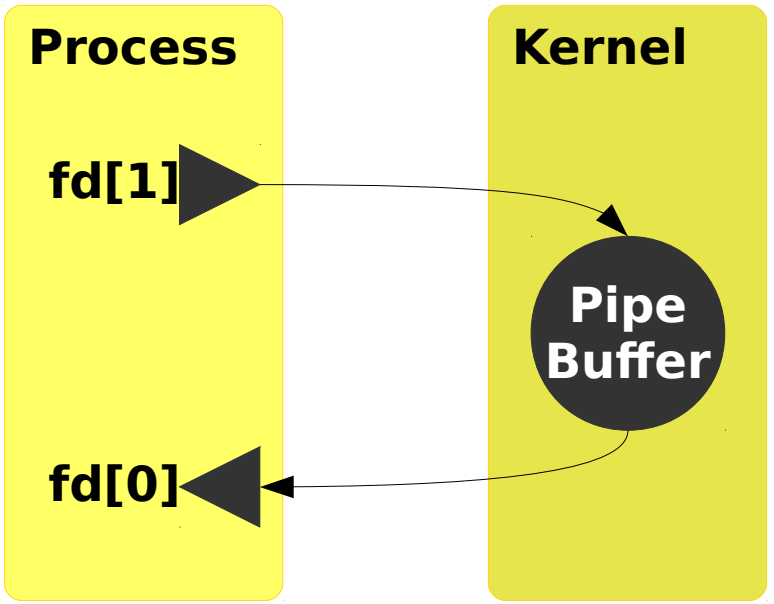


Inter Process Communications
Pipes – Direction of communication

- So it necessary to close one of the end form both sides



Inter Process Communications
Pipes – Working



Inter Process Communications
Pipes - Pros & Cons

PROS

- Naturally synchronized
- Simple to use and create
- No extra system calls required to communicate (read/write)

CONS

- Less memory size (4K)
- Only related process can communicate.
- Only two process can communicate
- One directional communication
- Kernel is involved

Inter Process Communications
Summary

- We have covered

Data exchange

Communication

- Pipes
- FIFO
- Shared memory
- Signals
- Sockets

Resource usage/access/control

Synchronization

- Semaphores

Stay Connected

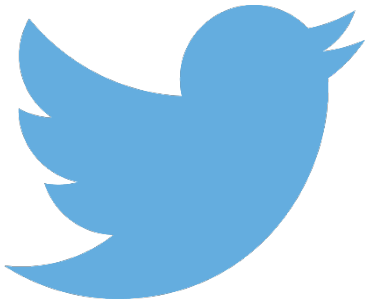
About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,

No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046

T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You