

# Comunicación entre procesos (IPC) - intro redes

Diego Fernandez Slezak<sup>1</sup>

<sup>1</sup>Departamento de Computación, FCEyN,  
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2024

## (2) La clase anterior...

- Vimos

## (2) La clase anterior...

- Vimos
  - El concepto de proceso en detalle.

## (2) La clase anterior...

- Vimos
  - El concepto de proceso en detalle.
  - Sus diferentes actividades.

## (2) La clase anterior...

- Vimos
  - El concepto de proceso en detalle.
  - Sus diferentes actividades.
  - Qué es una system call.

### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.

### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.

### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.
- Sirve para:



### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.
- Sirve para:
  - [Compartir información.](#)


### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.
- Sirve para:
  - Compartir información.
  - Mejorar la velocidad de procesamiento.

### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.
- Sirve para:
  - Compartir información.
  - Mejorar la velocidad de procesamiento.
  - Modularizar.

### (3) La de hoy

- Vamos a ver comunicaciones entre procesos.
- Ya sea entre procesos en un mismo equipo, o entre procesos remotos.
- Sirve para:
  - Compartir información.
  - Mejorar la velocidad de procesamiento.
  - Modularizar.
- La comunicación entre procesos suele llamarse *IPC*, por *InterProcess Communication*. 

## (4) Arquitectura multiproceso en Chrome

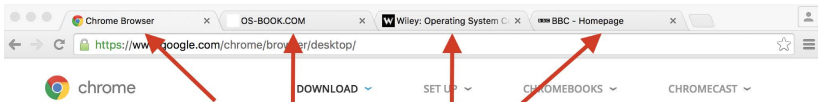
- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)

## (4) Arquitectura multiproceso en Chrome

- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)
  - Si hay problemas con un sitio, todo el browser se puede colgar.

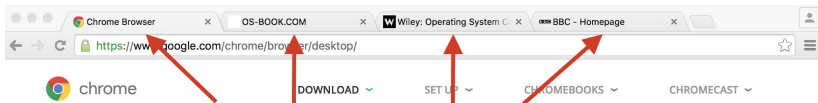
## (4) Arquitectura multiproceso en Chrome

- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)
  - Si hay problemas con un sitio, todo el browser se puede colgar.
- Chrome crea 3 tipos de procesos:



## (4) Arquitectura multiproceso en Chrome

- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)
  - Si hay problemas con un sitio, todo el browser se puede colgar.
- Chrome crea 3 tipos de procesos:
  - Browser: administra interface de usuario, acceso a disco y a red.

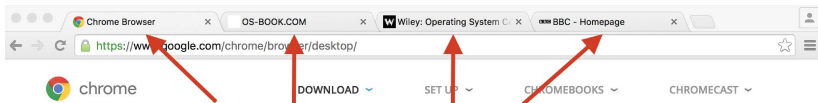


Each tab represents a separate process.



## (4) Arquitectura multiproceso en Chrome

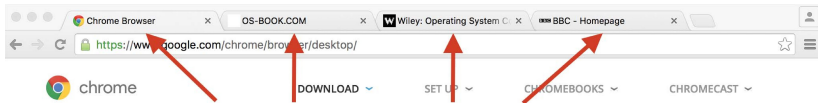
- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)
  - Si hay problemas con un sitio, todo el browser se puede colgar.
- Chrome crea 3 tipos de procesos:
  - Browser: administra interface de usuario, acceso a disco y a red.
  - Renderer: muestra las páginas, se ocupa del html y javascript. Uno nuevo por sitio. Corre en sandbox.



Each tab represents a separate process.

## (4) Arquitectura multiproceso en Chrome

- Muchos browsers corrían como un solo proceso (algunos aún lo hacen)
  - Si hay problemas con un sitio, todo el browser se puede colgar.
- Chrome crea 3 tipos de procesos:
  - Browser: administra interface de usuario, acceso a disco y a red.
  - Renderer: muestra las páginas, se ocupa del html y javascript. Uno nuevo por sitio. Corre en sandbox.
  - Plug-in: Proceso para cada tipo de plugin.



Each tab represents a separate process.

## (5) IPC

- Hay varias formas de IPC:

## (5) IPC

- Hay varias formas de IPC:
  - Memoria compartida.

## (5) IPC

- Hay varias formas de IPC:
  - Memoria compartida.
  - Algún otro recurso compartido (por ejemplo, archivo, base de datos, etc.).

## (5) IPC

- Hay varias formas de IPC:
  - Memoria compartida.
  - Algún otro recurso compartido (por ejemplo, archivo, base de datos, etc.).
  - Pipes

## (5) IPC

- Hay varias formas de IPC:
  - Memoria compartida.
  - Algún otro recurso compartido (por ejemplo, archivo, base de datos, etc.).
  - Pipes
  - Pasaje de mensajes (sockets)

## (5) IPC

- Hay varias formas de IPC:
  - Memoria compartida.
  - Algún otro recurso compartido (por ejemplo, archivo, base de datos, etc.).
  - Pipes
  - Pasaje de mensajes (sockets)
- Vamos a concentrarnos en el pasaje de mensajes.

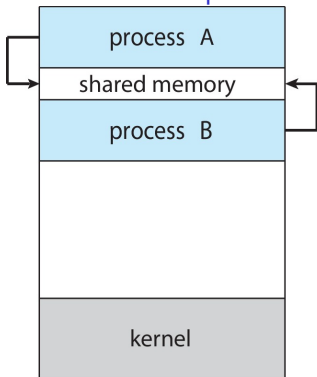


## (5) IPC

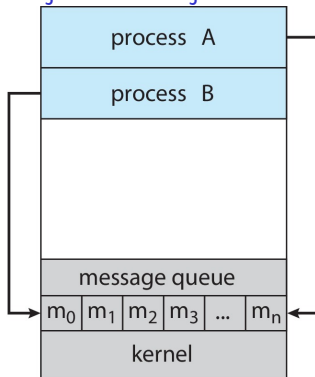
- Hay varias formas de IPC:
  - Memoria compartida.
  - Algún otro recurso compartido (por ejemplo, archivo, base de datos, etc.).
  - Pipes
  - Pasaje de mensajes (sockets)
- Vamos a concentrarnos en el pasaje de mensajes.
- En particular, entre procesos de la misma máquina (aunque esto no es obligatorio).

## (6) IPC

- a. Memoria compartida b. Pasaje de mensajes



(a)



(b)

## (7) IPC - Mem. compartida

```
from multiprocessing import Process, Value, Array

def f(n, a):
    n.value = 3.14
    for i in range(len(a)):
        a[i] = -a[i]

if __name__ == '__main__':
    num = Value('d', 0.0)
    arr = Array('i', range(5))

    p = Process(target=f, args=(num, arr))
    p.start()
    p.join()

    print(num.value)
    print(arr[:])
```

## (8) Pipes

- Un “pseudo archivo” que “esconde” una forma de IPC.

## (8) Pipes

- Un “pseudo archivo” que “esconde” una forma de IPC.
- Ordinary pipes:

## (8) Pipes

- Un “pseudo archivo” que “esconde” una forma de IPC.
- Ordinary pipes:
  - `ls -l | grep so`

## (8) Pipes

- Un “pseudo archivo” que “esconde” una forma de IPC.
- Ordinary pipes:
  - `ls -l | grep so`
- Named pipes:

## (8) Pipes

- Un “pseudo archivo” que “esconde” una forma de IPC.
- Ordinary pipes:
  - `ls -l | grep so`
- Named pipes:
  - `mkfifo -m 0640 /tmp/mituberia`



## (9) IPC - Pipes

```
from multiprocessing import Process, Pipe

def child_process(conn):
    conn.send("Hello from child process!")
    conn.close() # Close the connection when done

if __name__ == "__main__":
    parent_conn, child_conn = Pipe() # Create a pipe

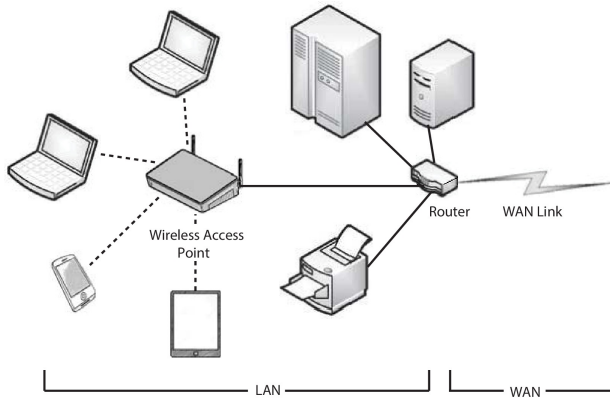
    p = Process(target=child_process, args=(child_conn,))

    p.start() # Start the child process

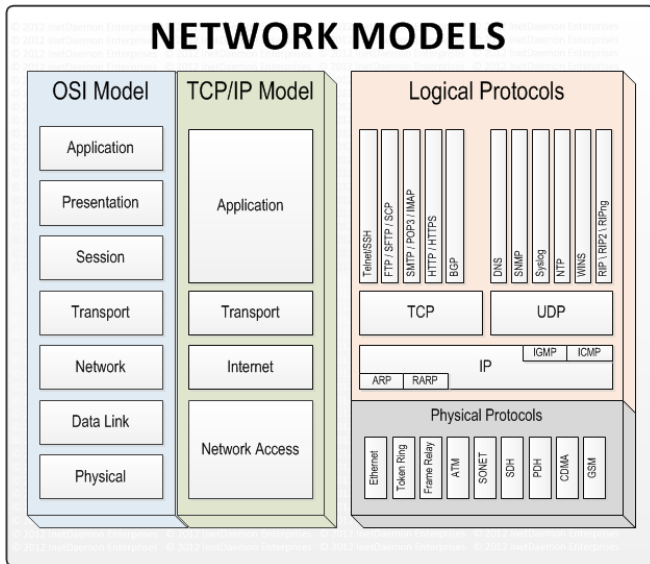
    print(parent_conn.recv()) # Receive message from child

    p.join() # Wait for the child process to finish
```

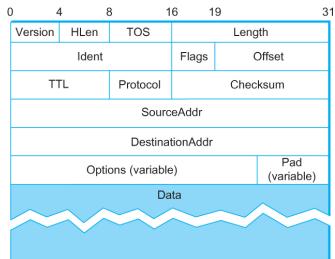
## (10) Redes LAN y WAN



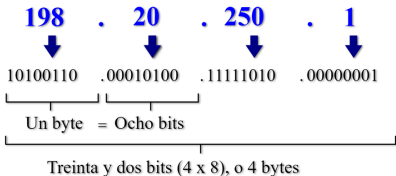
# (11) Stack TCP/IP



## (12) IPv4



Estructura de una dirección IPv4



Hay algunas direcciones especiales como 127.0.0.1 que es el equipo local (localhost).

## (13) TCP, UDP, ICMP

- TCP: Transmission Control Protocol. Protocolo orientado a la conexión. Provee control de flujo, recuperación de errores y confiabilidad.

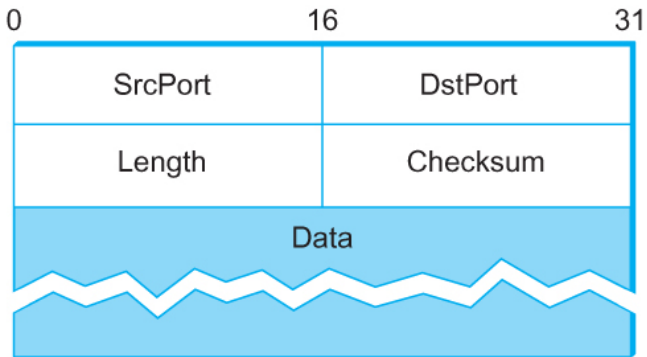
## (13) TCP, UDP, ICMP

- TCP: Transmission Control Protocol. Protocolo orientado a la conexión. Provee control de flujo, recuperación de errores y confiabilidad.
- UDP: User Datagram Protocol. Muy sencillo, no provee garantías. La recuperación de errores es responsabilidad de la aplicación.

## (13) TCP, UDP, ICMP

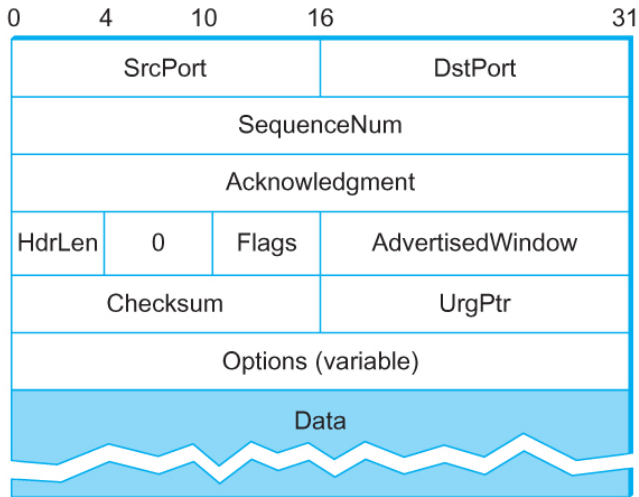
- TCP: Transmission Control Protocol. Protocolo orientado a la conexión. Provee control de flujo, recuperación de errores y confiabilidad.
- UDP: User Datagram Protocol. Muy sencillo, no provee garantías. La recuperación de errores es responsabilidad de la aplicación.
- ICMP: Internet Control Message Protocol. Es usado para mensajes de control, mensajes de error, etc. El Ping utiliza ICMP. Algunos tipos de ICMP: Echo request, Echo Reply, Destination Unreachable.

## (14) UDP





## (15) TCP



## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.

## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.
- Telnet: Viejo protocolo para acceder a una terminal en forma remota.

## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.
- Telnet: Viejo protocolo para acceder a una terminal en forma remota.
- SSH: Reemplazo del anterior. Resuelve problemas de seguridad y agrega funcionalidad.

## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.
- Telnet: Viejo protocolo para acceder a una terminal en forma remota.
- SSH: Reemplazo del anterior. Resuelve problemas de seguridad y agrega funcionalidad.
- SMTP: Envío de correo electrónico.

## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.
- Telnet: Viejo protocolo para acceder a una terminal en forma remota.
- SSH: Reemplazo del anterior. Resuelve problemas de seguridad y agrega funcionalidad.
- SMTP: Envío de correo electrónico.
- HTTP: Protocolo de comunicación utilizado por un browser para comunicarse con servidores web.

## (16) Protocolos de aplicación

- DNS: Permite resolver un nombre a una dirección IP. Ejemplo: `www.dc.uba.ar` es `157.92.27.128`.
- Telnet: Viejo protocolo para acceder a una terminal en forma remota.
- SSH: Reemplazo del anterior. Resuelve problemas de seguridad y agrega funcionalidad.
- SMTP: Envío de correo electrónico.
- HTTP: Protocolo de comunicación utilizado por un browser para comunicarse con servidores web.
- HTTPS: HTTP+TLS (permite aumentar seguridad en comunicaciones HTTP).

## (17) Sockets

- La idea de los sockets es la siguiente:



## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.

## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.
  - Luego de crearlo tengo que unirlo a algo, a una forma de comunicación concreta. Sería como conectar el enchufe a los cables.

## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.
  - Luego de crearlo tengo que unirlo a algo, a una forma de comunicación concreta. Sería como conectar el enchufe a los cables.
  - Una vez que tengo un socket conectado puedo leer y escribir de él como si fuese un dispositivo.


## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.
  - Luego de crearlo tengo que unirlo a algo, a una forma de comunicación concreta. Sería como conectar el enchufe a los cables.
  - Una vez que tengo un socket conectado puedo leer y escribir de él como si fuese un dispositivo.
  - Es decir, vale todo lo que vimos relacionado con bloqueo y scheduling.

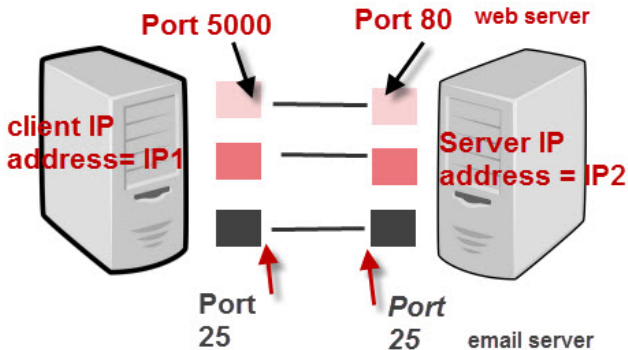
## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.
  - Luego de crearlo tengo que unirlo a algo, a una forma de comunicación concreta. Sería como conectar el enchufe a los cables.
  - Una vez que tengo un socket conectado puedo leer y escribir de él como si fuese un dispositivo.
  - Es decir, vale todo lo que vimos relacionado con bloqueo y scheduling.
  - Los detalles los vamos a ver en la práctica.

## (17) Sockets

- La idea de los sockets es la siguiente:
  - Un socket es el extremo de una comunicación. Piensen en el enchufe de la pared.
  - Luego de crearlo tengo que unirlo a algo, a una forma de comunicación concreta. Sería como conectar el enchufe a los cables.
  - Una vez que tengo un socket conectado puedo leer y escribir de él como si fuese un dispositivo.
  - Es decir, vale todo lo que vimos relacionado con bloqueo y scheduling.
  - Los detalles los vamos a ver en la práctica.
  - Lo importante es que para un proceso, hacer IPC es como hacer E/S. 


## (18) Sockets



IP Address + Port number = Socket


## TCP/IP Ports And Sockets

## (19) IPC sincrónico/asincrónico


- La comunicación entre procesos puede ser sincrónica o asincrónica: 




## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica


## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.


## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.


## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.
  - En general involucra bloqueo del emisor.


## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.
  - En general involucra bloqueo del emisor.
- Asincrónica


## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.
  - En general involucra bloqueo del emisor.
- Asincrónica
  - El emisor envía algo que el receptor va a recibir en algún otro momento.

## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.
  - En general involucra bloqueo del emisor.
- Asincrónica
  - El emisor envía algo que el receptor va a recibir en algún otro momento.
  - Requiere algún mecanismo adicional para saber si el mensaje llegó.

## (19) IPC sincrónico/asincrónico

- La comunicación entre procesos puede ser sincrónica o asincrónica: 
- Sincrónica
  - El emisor no termina de enviar hasta que el receptor no recibe.
  - Si el mensaje se envió sin error suele significar que también se recibió sin error.
  - En general involucra bloqueo del emisor.
- Asincrónica
  - El emisor envía algo que el receptor va a recibir en algún otro momento.
  - Requiere algún mecanismo adicional para saber si el mensaje llegó.
  - Libera al emisor para realizar otras tareas, no suele haber bloqueo, aunque puede haber un poco (por ejemplo, para copiar el mensaje a un buffer del SO).



## (20) Dónde estamos

- Vimos

- Vimos
  - Distintas formas de IPC.

- Vimos
  - Distintas formas de IPC.
  - Una breve introducción a conceptos de redes

- Vimos
  - Distintas formas de IPC.
  - Una breve introducción a conceptos de redes
- En la próxima teórica:

## (20) Dónde estamos

- Vimos
  - Distintas formas de IPC.
  - Una breve introducción a conceptos de redes
- En la próxima teórica:
  - *Vemos scheduling de procesos*