**ELO330 - Tarea #2**

# Tasas de Transferencia entre Mecanismos de Comunicación

Programación de Sistemas - 2022-2

*Tomás Velasquez Lee - Julio Contreras Fuica - Nicolás Canales Aravena*

---

# Documentación

## pfifo.c

*Sintaxis: pfifo <prefix_name><N>*, donde *N* es la cantidad de segundos que se ejecución y *prefix_name* un indicador para diferenciar sus recursos. Tiene como objetivo producir la información que será recibida por *cfifo.c*.

**Casos de fallo:** es importante que en caso de querer probar el programa de forma independiente, haya previamente ejecutado el programa *cfifo.c*.

## cfifo.c

*Sintaxis: cfifo <prefix_name><N>*, donde *N* es la cantidad de segundos que se ejecución y *prefix_name* un indicador para diferenciar sus recursos.

**Consideración:** es relevante mencionar que el cálculo de la tasa de transferencia se midió en Bytes. Esto con el fin de encontrar un equivalente en común con el oto medio de transmisión a comparar.

## pshmem.c

*Sintaxis: pshmem <prefix_name><N>*, donde *N* es la cantidad de segundos que se ejecución y *prefix_name* un indicador para diferenciar sus recursos.
Su funcionamiento se basa en escribir en el espacio de memoria compartida previamente creada por *cshmem.c*, con el objetivo de generar un tráfico de información entre procesos.
Hace uso de un espacio limitado de memoria el cual se va consumiendo circularmente, esto con el fin de solicitar el menor espacio posible a Aragorn.

**Casos de fallo:** es importante que en caso de querer probar el programa de forma independiente, haya previamente ejecutado el programa *cshmem.c*.

## cshmem.c

*Sintaxis: cshmem <prefix_name><N>* , donde *N* es la cantidad de segundos que se ejecución y *prefix_name* un indicador para diferenciar sus recursos.
Tiene como objetivo crear un espacio de memoria compartida para leer los datos del programa *pshmem.c*, a la vez que va analizando y guardando la cantidad de bytes leídos.

**Consideración:** es relevante mencionar que para que la comparación de tasas de bytes entre ambos sistemas de comunicación fuera justa, se decidió llevar sus resultados a bytes. Para este caso, se escribió, leyó y transmitió la información como variables del tipo *INT*, por lo que es equivalente a 4 Bytes.

### comparison.c

**Estrategia** Se generaron procesos hijos mediante la función `fork` para ejecutar, mediante `exec`, los programas necesarios para hacer la comparación. Estos procesos son gnuplot, cfifo, pfifo, cshmem y pshmem. **Consideración tomada:**
Fue necesario asegurar que los procesos hijos se ejecutaran en el orden correcto, es decir, primero los consumidores (que crean los recursos compartidos) y luego los productores que generan datos hacia los recursos creados.

## Gráficas y resultados:

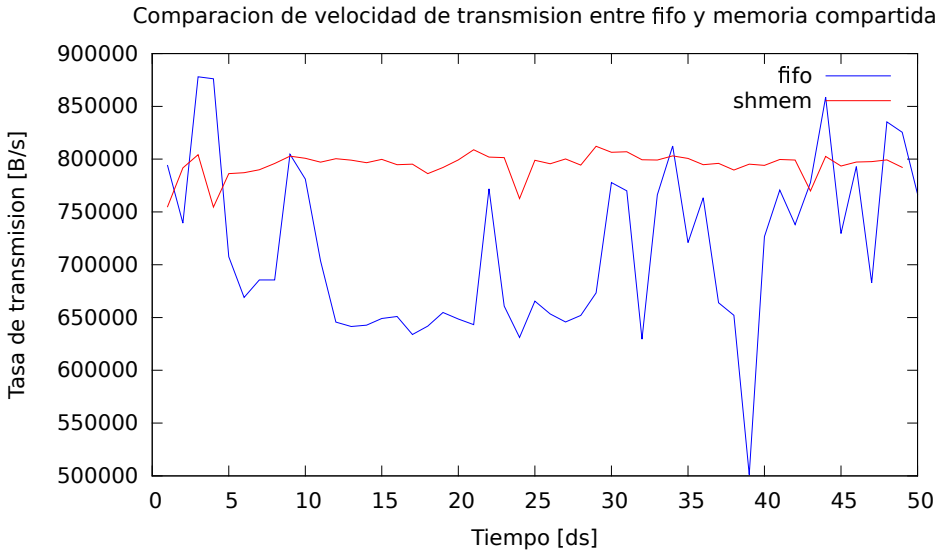


Figura 1: Gráfico resultante haciendo la prueba por 5 segundos.

De la imagen anterior es posible apreciar como en general la memoria compartida, en comparación con el uso de pipes, presenta un comportamiento mucho más estable a lo largo del tiempo, a la vez que también es el método más rápido. También es posible concluir que si bien por intuición el uso de pipes podría parecer más veloz, dado que funciona de sin bloqueo alguno, consumiendo lo enviado desde un programa a otro, la memoria compartida mediante el uso de semáforos, logra tener un muy buen manejo de recursos, asegurando la estabilidad y velocidad de los datos transferidos.

## Link del Respositorio:

`https://github.com/jcontrerasf/ELO330/tree/main/Tarea2`