

**TRABAJO PRÁCTICO**

**FINAL SISTEMAS OPERATIVOS:**

**“Inter Process Communication”**

**INTEGRANTES:**

* **Lucio Fernandez**
* **Nicolas Mongiello**
* **Pablo Cumpe**

Contenido

[Objetivo del trabajo práctico 2](#_Toc533009374)

[Introducción 2](#_Toc533009375)

[Explicación breve de la idea implementada 3](#_Toc533009376)

[Descripción de algunas partes del código 4](#_Toc533009377)

[Conclusión 4](#_Toc533009378)

## Objetivo del trabajo práctico

El trabajo práctico consiste en aprender a utilizar los distintos tipos de IPCs presentes en un sistema POSIX. Para ello se implementará un sistema que distribuirá tareas de cálculo pesadas entre varios pares.

## Introducción

El siguiente trabajo práctico fue implementado en Linux con lenguaje C. El desarrollo del mismo implicó una investigación sobre los siguientes temas:

* PIPES
* NAMED PIPE
* SEGMENTOS DE MEMORIA COMPARTIDA
* SEMAFOROS
* CREACION Y USO DE HILOS
* CREACION Y USO DE PROCESOS HIJOS
* SEÑALES
* HASH MD5
* MANEJO DE ARCHIVOS Y DIRECTORIOS
* CREACION DE TEST PARA PRUEBAS

## Explicación breve de la idea implementada

El trabajo consiste en dos procesos:

El **proceso padre** recibe por parámetro:

* La dirección de un directorio utilizando todos los archivos que se encuentren en el.
* La ruta donde se escribirá el archivo final de salida con los resultados generados.

La función del proceso padre es distribuir los archivos en N procesos hijos para su procesamiento.

La comunicación entre el proceso padre con sus hijos y viceversa, se realiza mediante pipes.

Cada proceso hijo tiene como objetivo hashear cada archivo enviado por el proceso padre usando el algoritmo md5 y luego el resultado es devuelto al padre.

El proceso padre luego de generar todos los hijos, dispara dos hilos:

* El primer hilo agrega a un sector de memoria compartida el resultado del procesamiento de los hijos y también escribe lo escribe en el archivo de salida.
* El segundo hilo queda a la espera de que aparezca un proceso externo llamado “vista”, al cual le comparte mediante un named pipe el segmento de memoria compartida .

El **proceso vista** imprime por pantalla los resultados obtenidos de la memoria compartida.

Se utilizan semáforos para escribir y leer de la memoria compartida.

Una vez finalizados los dos hilos , finaliza el **proceso padre**.

Observaciones

* Pudimos observar que en caso de que se llene el sector de memoria compartida el **proceso padre** queda en “stand by” hasta que se libere, es decir que aparezca el **proceso vista** y empiece a consumir y liberar la memoria.
* Se genero un script para generar N archivos de distinto tamaño, a mayor tamaño del archivo mayor es el tiempo en procesarlos.

## Descripción de algunas partes del código

Cálculo Hash MD5

Teniendo en cuenta la premisa del trabajo práctico para éste punto, empezamos a investigar y vimos que existía la posibilidad de ejecutar un comando de sistema operativo y capturar lo devuelto a través de la salida estándar de comandos.

Esto lo hicimos con la función popen() que recibe por parámetro el comando y el tipo de operación (lectura o escritura). Este comando crea un pipe, crea un nuevo proceso hijo e invoca al núcleo. Una vez ejecutado el comando, la función popen() (en modo lectura) vuelca el resultado a la salida estándar de comandos.

De esta manera, resolvimos el calculo del hash definiendo una función que ejecute el comando md5sum a través de la función popen() y leyendo el resultado de la línea de comandos.

Test de pablo

## Conclusión