Sistemas Operativos Práctica 4: Implementación de pipes para llamadas al sistema

Nicolás Sela Ramos
Arturo Requejo Portilla

UNIVERSIDAD NEBRIJA

Tabla de contenido

I. Enunciados

A. Ejercicio 1

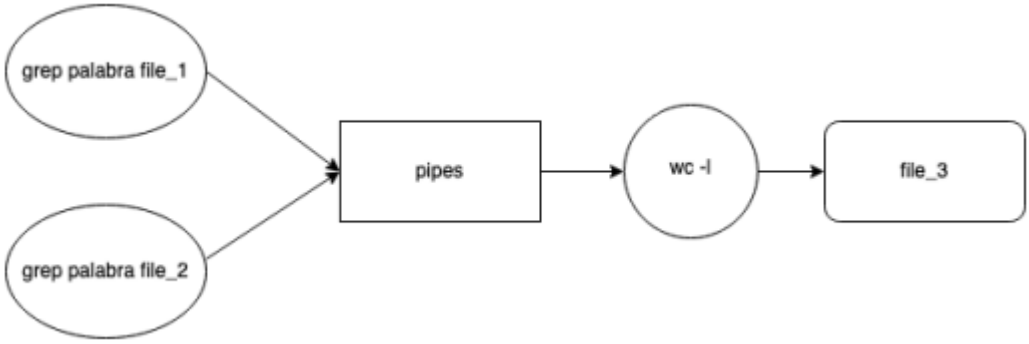
Escribir un programa que genere 3 procesos en paralelo y que colaboran de forma conjunta para realizar la siguiente tarea: **Justificar el proceso 10 PUNTOS.**

1. El primer proceso, utiliza el comando `grep` para encontrar en el (file_1), la palabra que se pasa por la línea de comandos.
2. EL segundo proceso, utiliza el comando `grep` para encontrar en el (file_2)), la palabra que se pasa por la línea de comandos.
3. El tercer proceso, utilizando el comando `wc -l` leerá por el pipe el número de coincidencias en los ficheros (file_1, file_2) y los escribirá en el (file_3).

La instrucción de ejecución será la siguiente:

```
./ejecutable palabra file_1 file_2 file_3
```

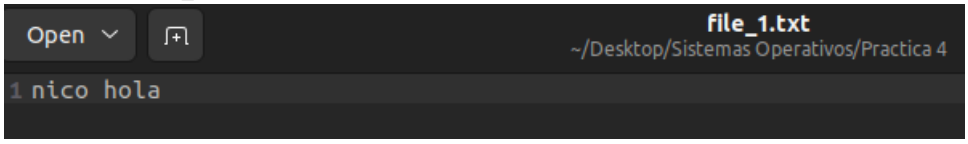
En la siguiente figura, se muestra gráficamente la comunicación requerida entre procesos:



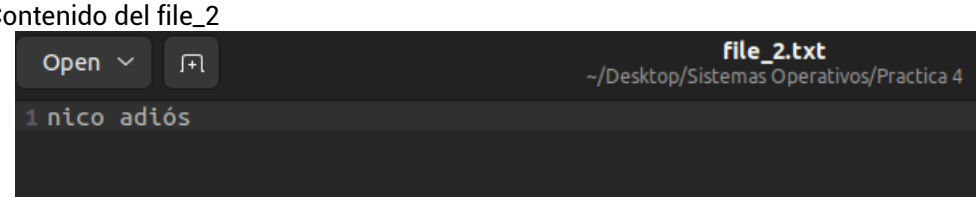
II. Resultado

A. Ejercicio 1

Contenido del file_1

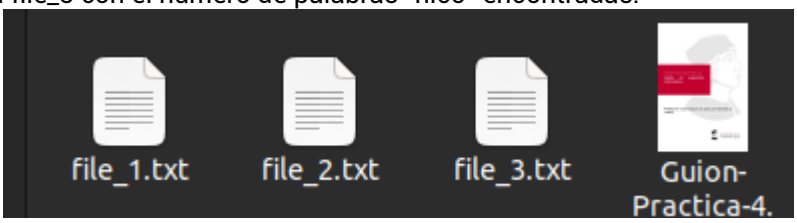


Contenido del file_2



Ejecutamos el comando

Se crea el file_3 con el número de palabras" nico" encontradas:



Creamos un único pipe que sigue un formato unidireccional. Donde solo se podrán ejecutar una operación de escritura o lectura. Asegurando así que por un extremo leamos y por el otro escribamos.

Posteriormente creamos el primer proceso, este proceso contará la palabra que le pasemos por argumento del fichero 1. Para ello empezamos realizando una llamada al sistema (dup2) abriendo así el archivo, posteriormente con `STDOUT_FILENO` realizamos una acción de lectura leyendo el archivo de nombre introducido por terminal y lo mandamos por el pipe 1 (lectura). Luego cerramos los canales del pipe y por último mediante la función `execlp` ejecutamos el comando grep por la terminal donde buscamos la palabra, argumento [1] del archivo pasado en el argumento [2].
Para el proceso 2 se realiza la misma ejecución que en el proceso 1 pero en el archivo 2.

Para el 3 y último proceso empezamos creando un fichero de nombre del último argumento pasado por terminal con permisos de lectura y escritura para el propietario del archivo y de lectura para los grupos y otros usuarios. Luego realizando otra llamada al sistema abrimos el canal del pipe de escritura y creamos el archivo. Luego cerramos el pipe para ambos modos y por último mediante `wc` escribimos el resultado en el archivo.

Mediante un único pipe comunicamos el resultado de la lectura de todos los archivos de forma paralela independiente del número de procesos. Con esto conseguimos una comunicación paralela entre, en este caso, 2 procesos con 1, donde este 1 recibe la información y escribe un archivo con los resultados

B. Conclusiones

En esta práctica hemos realizado una implementación más avanzada de los pipes usándolos para hacer llamadas al sistema, es decir, ejecutar comandos de un sistema operativo UNIX, transferir información de archivos y conseguir una comunicación paralela entre un proceso y varios.

C. Bibliografía

C Read File From Command Line With Code Examples - https://www.folkstalk.com/tech/c-read-file-from-command-line-with-code-examples/