Iván Álvarez García y Andrea Gómez Bobes

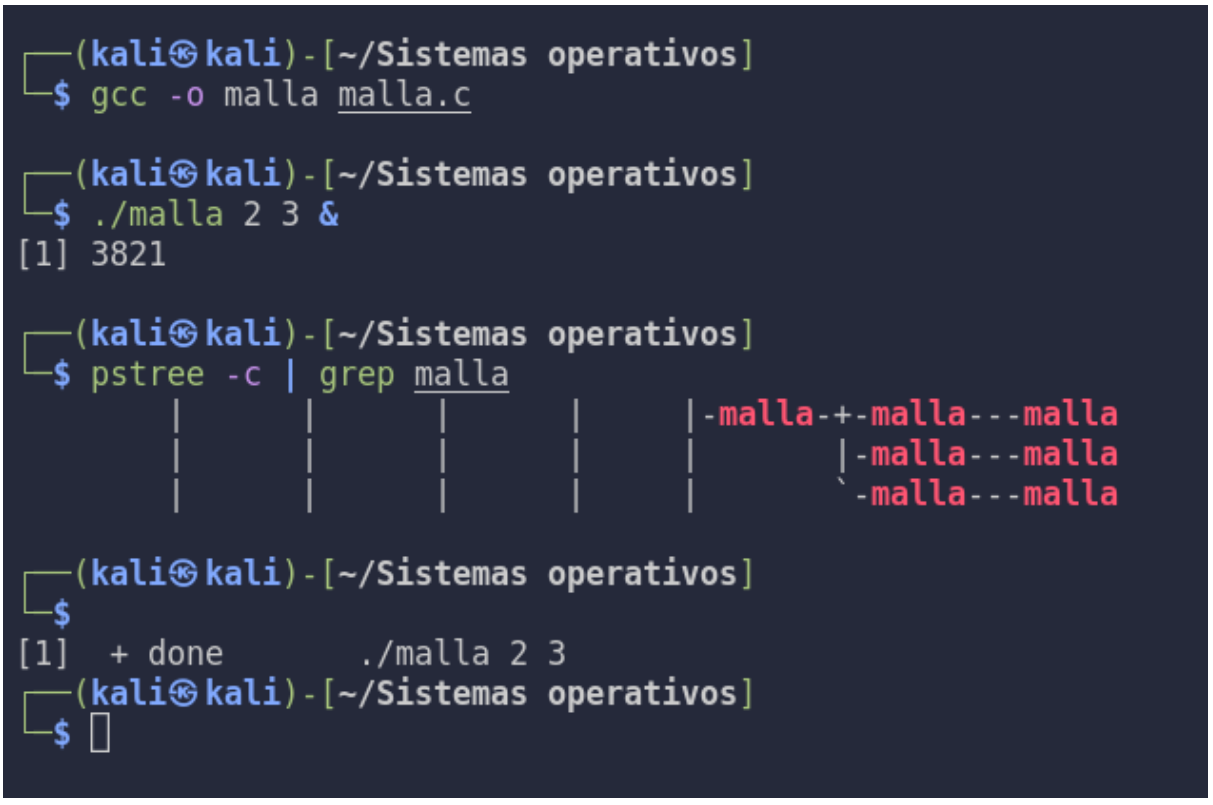
Práctica 1:
Gestión de procesos y archivos. Comunicación entre procesos: tuberías y memoria compartida

Ejercicio 1:

a) Realiza un programa llamado malla.c que produzca el siguiente árbol de procesos. El programa recibirá dos argumentos 'x' e 'y' que representan el número de filas y columnas. Para comprobar la estructura de procesos que realmente estamos creando debemos emplear la llamada al sistema "pstree –c".

Para este apartado hemos controlado primeramente el número de argumentos pasados al programa. Luego creamos un bucle *for* para crear tantos procesos como número de columnas y otro bucle del mismo tipo para crear tantos procesos como filas hay. En ambos bucles se controlan qué procesos realizan cada bucle.

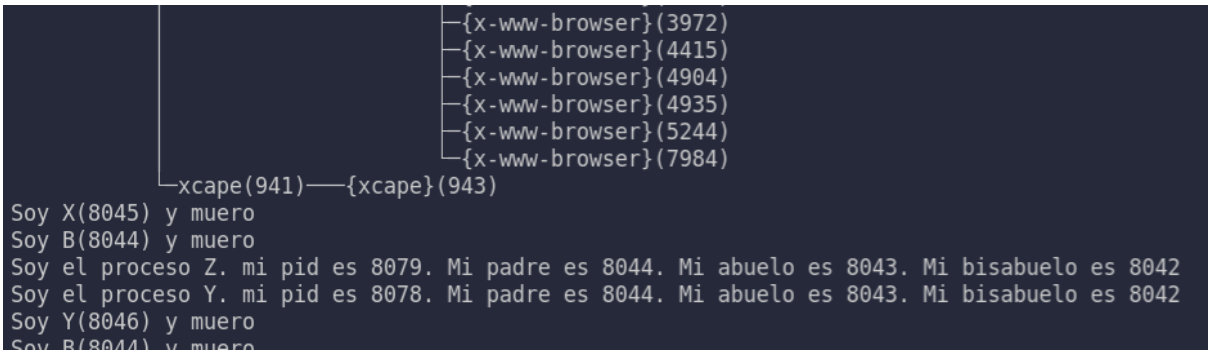
Por ejemplo, aquí se crea un árbol con 2 filas y 3 columnas:



b) Realiza un programa llamado ejec.c que reciba un argumento. El programa tendrá que generar el árbol de procesos que se indica y llevar a cabo la funcionalidad que se describe a continuación. El proceso Z, transcurridos los segundos indicados por el argumento, ejecutará
el comando "pstree". El proceso Z no puede utilizar el comando "sleep", por lo que el proceso Z debe planificarse una alarma con los segundos indicados por el argumento. Se deberá controlar la correcta destrucción del árbol (los padres no pueden morir antes que los hijos).

A pesar de que este ejercicio no funciona correctamente, el proceso a seguir sería hacer la estructura de árbol con los procesos y programar un *timer* para que pasados los segundos indicados por argumento los procesos mueran en orden.

Aún así, aquí hay un ejemplo de ejecución de nuestro programa:



Como podemos ver, los procesos no mueren en el orden correcto.

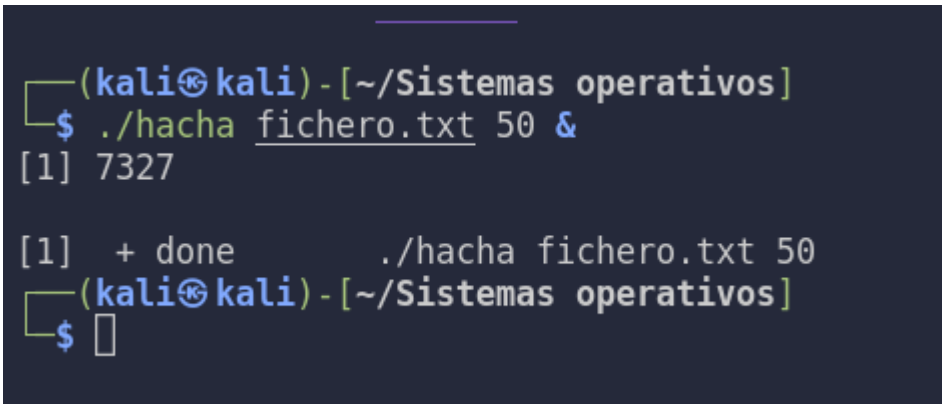
Ejercicio 2:

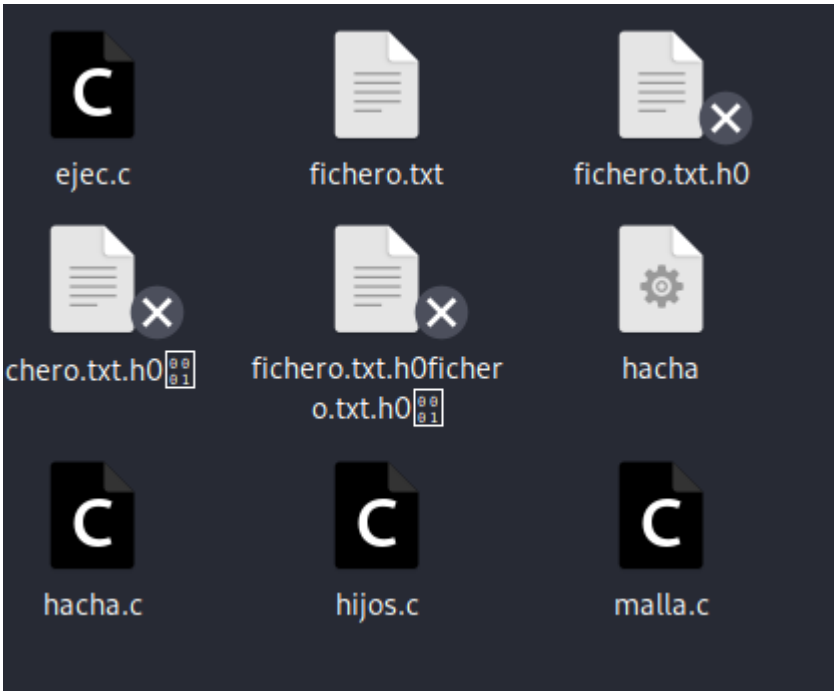
Realizar un programa llamado hacha.c que divida un archivo en varios trozos (archivos) con el mismo nombre y extensión h00, h01, … teniendo en cuenta el formato y las siguientes consideraciones:
$ hacha <archivo> <tamaño>
archivo a dividir
tamaño en bytes de los archivos divididos

Para dividir, el proceso hacha (proceso padre) generará tantos procesos hijos como archivos se tengan que crear. Mediante tuberías el proceso padre enviará la información a los hijos (a cada hijo su trozo) y serán estos últimos los que creen los archivos de destino y escriban en ellos.

Primero tenemos que saber el número de bytes que ocupa el archivo a elegir. Para ello utilizamos la función *stat()*. Luego dividimos el tamaño del archivo por el argumento pasado y el resultado será el número de procesos hijos a crear. Por último hacemos que cada hijo cree un archivo y reciba la información por *pipes* de parte del proceso padre.

Aquí tenemos un ejemplo de ejecución:





Ejercicio 3:

Diseña un programa llamado hijos.c que cree un árbol de procesos según la siguiente estructura a partir de dos parámetros.

Para este ejercicio debemos controlar de nuevo lo que hacen los procesos hijos. Primero controlamos que solamente el que sea hijo cree un nuevo proceso, tantas veces como el parámetro x indique. Seguidamente, indicamos al último proceso creado que cree tantos hijos como el parámetro y indica.

En el siguiente ejemplo de ejecución ponemos x = 2 e y = 4. Además añadimos comentarios refiriéndose los procesos a sus hijos para saber que es correcto.

```
┌──(kali㉿kali)-[~/Sistemas operativos]
└─$ gcc -o hijos hijos.c

┌──(kali㉿kali)-[~/Sistemas operativos]
└─$ ./hijos 2 4 &
[1] 4276

Soy el subhijo 4278 mis padres son: 4276
Soy el subhijo 4279 mis padres son: 4276,4278
┌──(kali㉿kali)-[~/Sistemas operativos]
└─$ pstree -c | grep hijos                                                    1 ⚙
        |       |       |       |     |-hijos---hijos---hijos-+-hijos
        |       |       |       |     |                       |-hijos
        |       |       |       |     |                       |-hijos
        |       |       |       |     |                       `-hijos

┌──(kali㉿kali)-[~/Sistemas operativos]
└─$ Soy el superpadre (4276) mis hijos finales son: 4278, 4279, 4280, 4281, 4282, 4
283

[1]  + done       ./hijos 2 4
┌──(kali㉿kali)-[~/Sistemas operativos]
└─$
```