UNIVERSIDAD NACIONAL

SAN AGUSTIN DE AREQUIPA

Facultad de Ingeniería, Producción y Servicios

Escuela Profesional de Ciencia de la Computación

Laboratorio 4

Presentado por:

Parizaca Mozo, Paul Antony

CUI:

20210686

Curso:

Sistemas Operativos - Grupo A

Github:

https://github.com/PaulParizacaMozo/SistemasOperativos

Docente:

Yessenia Yari Ramos

Arequipa, Perú

2023

EJERCICIO 1

El archivo ejercicio1.c y ejercicio1-2.c se encuentran comentados en el siguiente link: https://github.com/PaulParizacaMozo/SistemasOperativos/tree/main/Laboratorio4

1. Explicar qué está haciendo el código (se puede comentar dentro del archivo)

Define dos variables que serán variables (i,j) además la variable rf que almacenará el identificador del respectivo proceso.
Al hacer uso de la función fork creamos un proceso hijo.
Ambos procesos pasarán por un switch, usando la variable rf que almacena el identificador si es -1 imprime un mensaje que indica que hubo un error al crear el proceso hijo, si es igual a 0 es el proceso hijo e asigna un valor a su variable i=0 la cual en un bucle que itera 5 veces hará que incremente de 2 en 2 e imprime su valor. Si el valor es distinto de 0 o 1 significa que se está ejecutando el proceso padre el cual asigna a su variable i = 1 y luego en un bucle que itera 5 veces incrementa en 2 ese valor y lo imprime.
Para terminar el programa ambos procesos imprimen sus identificadores.

2. ¿Son las variables enteras “i” y “j” del proceso padre las mismas que las del proceso hijo?

No son las mismas variables, como bien sabemos fork() crea una copia de la memoria del proceso padre, pero estas son independientes,por eso cada proceso ya sea el hijo o el padre tiene sus propias variable i,j que son independientes de las otras.

3. Cambia el código de “ejercicio1.c” para que ambos procesos inicien de una variable “i” con igual valor, pero uno la incremente de uno en uno y el otro de dos en dos. Guardar el nuevo programa como el archivo “ejercicio1-2.c”.

EJERCICIO 2

Código comentado del ejercicio2.c en el siguiente link:
https://github.com/PaulParizacaMozo/SistemasOperativos/tree/main/Laboratorio4

1. La expresión “fd1 = creat(“ficheroA”, 0666)” que está realizando?¿Qué significado tiene la constante “0666”? ¿Qué permisos tienen los dos ficheros, “ficheroA” y “ficheroB”, tras la ejecución del “ejercicio2.c”?

fd1 = creat(“ficheroA”, 0666)

Crea el fichero llamado ficheroA y asigna su descriptores(ident) a la variable fd1,
en el primer argumento le indica el nombre del fichero y en el segundo argumento especifica los permisos del fichero a crear.

0666: Indica que el fichero se creará con permisos de lectura y escritura.
lo mismo sucede con el ficheroB.
En resumen se crean los ficheros: ficheroA y ficheroB con permisos de lectura y escritura.

Ejecución del archivo

```
❯ gcc -Wall -o eject ejercicio2.c && ./eject

Final de ejecucion de 24349

Final de ejecucion de 24350
❯ ls fichero*
ficheroA  ficheroB

~/UNSA/SistemasOperativos/Laboratorio4 on  main !2 ?7
```

Si listamos los ficheros que comiencen con fichero vemos que se crearon los dos ficheros correctamente

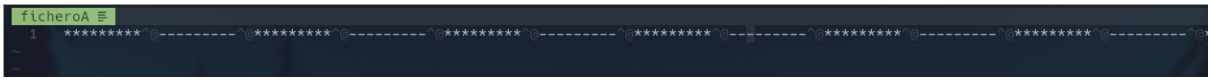
2. La ejecución concurrente de las escrituras de los procesos padre e hijo da lugar a que las cadenas “********“ y “----------“ las cuales están alternadas en los ficheros resultantes. Modifica “ejercicio2.c” para que, mediante la utilización de la función “sleep()”, la frecuencia a la que el proceso hijo escribe en los ficheros sea menor que la del proceso padre, es decir, que realice menos escrituras por unidad de tiempo. ¿En qué afecta eso al contenido de los ficheros? Guarda el nuevo programa como el archivo usleep“ejercicio2-1.c”**

La función usleep() permite que otros procesos se ejecuten mientras el proceso actual que llamá a la función usleep() espero un periodo de tiempo.

Al hacer que un proceso espere un periodo de tiempo mayor al otro, las instrucciones que realiza, en este caso escribir en el archivo, se realizan antes que las que tienen un tiempo de espera mayor.

Luego de ejecutar el ejercicio2.c

-ficheroA

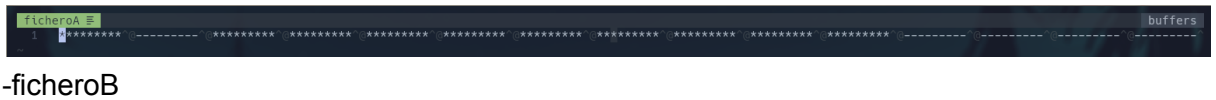


-ficheroB

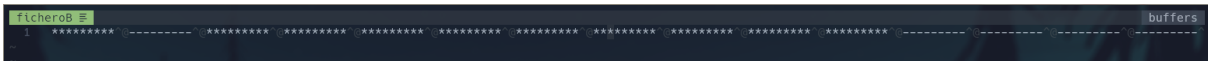
Ejecutamos el nuevo fichero ejercicio2-1.c que tiene los tiempos de espera modificados

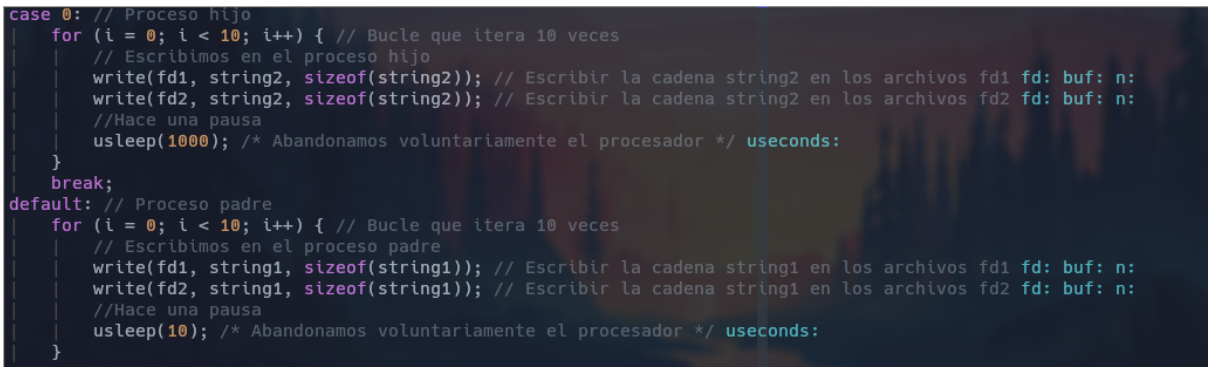
```
❯ gcc -Wall -o eject ejercicio2-1.c && ./eject

Final de ejecucion de 24301

Final de ejecucion de 24302
❯ ls fichero*
ficheroA  ficheroB

~/UNSA/SistemasOperativos/Laboratorio4 on  main !2 ?7
```

-ficheroA



-ficheroB





```
case 0: // Proceso hijo
    for (i = 0; i < 10; i++) { // Bucle que itera 10 veces
        // Escribimos en el proceso hijo
        write(fd1, string2, sizeof(string2)); // Escribir la cadena string2 en los archivos fd1 fd: buf: n:
        write(fd2, string2, sizeof(string2)); // Escribir la cadena string2 en los archivos fd2 fd: buf: n:
        //Hace una pausa
        usleep(1000); /* Abandonamos voluntariamente el procesador */ useconds:
    }
    break;
default: // Proceso padre
    for (i = 0; i < 10; i++) { // Bucle que itera 10 veces
        // Escribimos en el proceso padre
        write(fd1, string1, sizeof(string1)); // Escribir la cadena string1 en los archivos fd1 fd: buf: n:
        write(fd2, string1, sizeof(string1)); // Escribir la cadena string1 en los archivos fd2 fd: buf: n:
        //Hace una pausa
        usleep(10); /* Abandonamos voluntariamente el procesador */ useconds:
    }
```

Como podemos ver en el código para el proceso hijo pusimos una espera de 1000 microsegundos en cambio para el proceso padre una espera de 10 microsegundos

Por ello cuando vemos los nuevos ficheros creados vemos que el patrón en el cual se escriben las cadenas son distintos a la ejecución del ejercicio2.c . En este caso primero se escribe la cadena string1 y luego la cadena string2 hasta ahí todo bien, pero en ese momento ambos procesos se encuentran con un usleep() distinto para el proceso padre es un usleep(10) en cambio para el proceso hijo es un usleep(1000) por ello es que vemos que en los ficheros se escriben de forma seguida las cadenas string1 y despues recien se escriben las cadenas string2 esto se debe a que el proceso padre tuvo que esperar un menor tiempo que el proceso hijo.