

**“UNIVERSIDAD NACIONAL DE SAN AGUSTÍN”  
FACULTAD DE INGENIERÍA PRODUCCIÓN Y SERVICIOS  
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN**



**CURSO:**

ESTRUCTURA DE DATOS AVANZADOS

**TEMA:**

PRÁCTICA 02

**DOCENTE:**

YESSENIA YARI

**AUTOR:**

JAVIER WILBER QUISPE ROJAS

**Arequipa – Perú**

**2023**

## PRACTICA 2

### 1. Explicar que esta haciendo el codigo (se puede comentar dentro del archivo)

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int j;
    pid_t rf; // Declaración de una variable para almacenar el resultado de
    fork()

    rf = fork(); // Crear un proceso hijo duplicando el proceso actual

    switch (rf) // Se utiliza una estructura switch para manejar el resultado de
    fork().
    {
        case -1: //Si fork() devuelve -1, significa que hubo un error al crear
        el proceso hijo, y se muestra un mensaje de error.

            printf("\nNo he podido crear el proceso hijo");
            break;
        case 0:
            // fork() devuelva 0, significa que el código se está ejecutando en
            el proceso hijo. Se inicializa la variable i en 0 y se muestra información que
            identifica al proceso hijo, incluyendo su PID y el valor inicial de i.
            i = 0;
            printf("\nSoy el hijo, mi PID es %d y mi variable i (inicialmente a
            %d) es par", getpid(), i);
            for (j = 0; j < 5; j++) { // fork() devuelva 0, significa que el
            código se está ejecutando en el proceso hijo. Se inicializa la variable i en 0 y
            se muestra información que identifica al proceso hijo, incluyendo su PID y el
            valor inicial de i.
                i++;
                i++;
                printf("\nSoy el hijo, mi variable i es %d", i);
            };
            break;
        default:
            // Este código se ejecuta en el proceso padre
            i = 1;
            printf("\nSoy el padre, mi PID es %d y mi variable i (inicialmente a
            %d) es impar", getpid(), i);
            for (j = 0; j < 5; j++) {
                i++;
                i++;
                printf("\nSoy el padre, mi variable i es %d", i);
            };
    };

    printf("\nFinal de ejecucion de %d \n", getpid());
    exit(0);
}
```

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  
“ejercicio2-1.c”

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>

int main() {
    int i;
    int fd1, fd2;
    const char string1[10] = "*****";
    const char string2[10] = "-----";
    pid_t rf;

    fd1 = creat("ficheroA", 0666);
    fd2 = creat("ficheroB", 0666);
    rf = fork();

    switch (rf) {
        case -1:
            printf("\nNo he podido crear el proceso hijo");
            break;
        case 0:
            // Este código se ejecuta en el proceso hijo
            for (i = 0; i < 10; i++) {
                sleep(1); // Pausa de 1 segundo
                write(fd1, string2, sizeof(string2));
                write(fd2, string2, sizeof(string2));
            }
            break;
        default:
            // Este código se ejecuta en el proceso padre
            for (i = 0; i < 10; i++) {
                write(fd1, string1, sizeof(string1));
                write(fd2, string1, sizeof(string1));
            }
    };

    printf("\nFinal de ejecución de %d\n", getpid());
    exit(0);
}
```