

Universidad Nacional de San Agustín de Arequipa Escuela Profesional de Ciencia de la Computación

Curso: SISTEMAS OPERATIVOS



"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



Universidad Nacional de San Agustín de Arequipa

Escuela Profesional de Ciencia de la Computación Curso: SISTEMAS OPERATIVOS



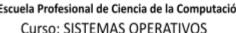
PRACTICA 2

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

```
int main() {
    int j;
   pid_t rf; // Declaración de una variable para almacenar el resultado de
    rf = fork(); // Crear un proceso hijo duplicando el proceso actual
   switch (rf) // Se utiliza una estructura switch para manejar el resultado de
    {
           printf("\nNo he podido crear el proceso hijo");
        case 0:
           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
                i++;
               printf("\nSoy el hijo, mi variable i es %d", i);
           };
           break;
            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++;
               printf("\nSoy el padre, mi variable i es %d", i);
           };
   printf("\nFinal de ejecucion de %d \n", getpid());
   exit(0);
}
```



Universidad Nacional de San Agustín de Arequipa Escuela Profesional de Ciencia de la Computación





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

No, las variables "i" y "j" del proceso padre no son las mismas que las del proceso hijo. Cada proceso tiene sus propias copias independientes de estas variables.

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".

ejercicio2.c

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"?

La expresión fd1 = creat("ficheroA", 0666) está creando un nuevo archivo llamado "ficheroA" en el directorio actual. La función creat es una llamada al sistema que crea un nuevo archivo y devuelve un descriptor de archivo (file descriptor), que es un número que se puede utilizar para realizar operaciones de lectura y escritura en ese archivo. En este caso, fd1 se convierte en el descriptor de archivo para el archivo "ficheroA".

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,



Universidad Nacional de San Agustín de Arequipa

Escuela Profesional de Ciencia de la Computación Curso: SISTEMAS OPERATIVOS



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"

```
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) {
            printf("\nNo he podido crear 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;
            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);
```