UA

ProgramaciónConcurrente

DanielAsensiRochDNI : **48776120C**

24 de noviembre de 2022

Índice

1. Lectores/Escritores con prioridad de lectura

1.1. Descripción del problema

Deberemos crear 10 hilos lectores, los cuales leeran 10 recursos cada uno, y 5 hilos escritores, los cuales modificarán un recurso 5 veces cada uno, este recurso será una variable entera la cual se inicializará con el valor -1. Características del problema:

- **Lectores** : desean leer un recurso, dos o más pueden acceder simultáneamente al recurso
- **Escritores** : actualizan la información del recurso, sólo uno puede acceder al recurso: acceso exclusivo al recurso

Variables y semáforos a utilizar:

- **mutex** : controla el acceso en exclusión mutua a la variable compartida readers y sirve de barrera para los escritores.
- **lectores** : funciona como un semáforo de exclusión mutua para los escritores, también lo utiliza el primer/último lector para entrar/salir de la sección crítica, pero no será utilizada mientras haya otros lectores o escritores en la sección crítica
- **escritores** : número de lectores en la sección crítica

1.2. Resolución del problema

1.2.1. Inicialización de variables y semáforos

Lo primero que deberemos realizar será inicializar nuestros semáforos y variables para definir la cantidad de lectores, escritores, así como los semaforos y el recurso

```
// Variables globales para el número de lectores y escritores
int LECTORES = 10;
int ESCRITORES = 5;

// Semaforos
sem_t mutex;
sem_t writer;

// nº lectores en la seccion critica
int readers = 0;

// recurso
int recurso = -1;
```

1.2.2. Bloqueo y desbloqueo de lector

Logica de bloqueo de la sección critica para bloquear la sección critica para escritores

```
// Prioridad lector
void bloqueoLector()
{
    sem_wait(&mutex);
    readers++;

    if (readers == 1)
    {
        sem_wait(&writer);
```

```
    }
    sem_post(&mutex);
}

void desbloqueoLector()
{
    sem_wait(&mutex);
    readers--;

    if (readers == 0)
    {
        sem_post(&writer);
    }
    sem_post(&mutex);
}
```

1.2.3. Definimos la función del lector y el escritor

Se define el comportamiento de los lectores y los escritores

```
// Funciones lector-escritor
void *lector(void *id)
{

    for (int i = 0; i < LECTORES; i++)
    {
        bloqueoLector();
        printf("El lector %d ha leido un valor de %d\n", *(int *)id, recurso);
        desbloqueoLector();
        sleep(1);
    }
    pthread_exit(id);
}

// Funcion del escritor
void *escritor(void *id)
{

    for (int i = 0; i < ESCRITORES; i++)
    {
        sem_wait(&writer);
        printf("El escritor %d ha actualizado el recurso\n", *(int *)id);
        recurso = *(int *)id; // modificar el recurso
        sem_post(&writer);
        sleep(1);
    }
    pthread_exit(id);
}
```

1.2.4. Creación y finalización de hilos

Estas dos funciones se encargarán de la creación de los hilos indistintamente de si son lectores o escritores.

```
void creacionHilos(pthread_t hilos[], int id[], int num, void *(*func)(void *))
{
```

```
    for (int i = 0; i < num; i++)
    {
        id[i] = i;
        int error = pthread_create(&hilos[i], NULL, func, &id[i]);
        if (error)
        {
            fprintf(stderr, "Error: %d: %s\n", error, strerror(error));
            exit(-1);
        }
    }
}

void finalizarHilos(pthread_t hilos[], int num)
{

    int *salida;

    for (int i = 0; i < num; i++)
    {
        int error = pthread_join(hilos[i], (void **)&salida);
        if (error)
        {
            fprintf(stderr, "Error: %d: %s\n", error, strerror(error));
        }
    }
}
```

1.2.5. Main

Nuestro main se encargará de llamar a la funciones previamente definidas así de como inicializar los vectores de los hilos y los ids de los mismos, tanto como de lectores como de escritores.

```
int main()
{
    pthread_t tidlector[LECTORES];
    pthread_t tidescritor[ESCRITORES];
    int idsLectores[LECTORES];
    int idEscritores[ESCRITORES];

    // Inicializacion de semaforos
    sem_init(&mutex, 0, 1);
    sem_init(&writer, 0, 1);

    // Se crean los hilos
    creacionHilos(tidescritor, idEscritores, ESCRITORES, escritor);
    creacionHilos(tidlector, idsLectores, LECTORES, lector);

    // Se espera a que los hilos terminen
    finalizarHilos(tidlector, LECTORES);
    finalizarHilos(tidescritor, ESCRITORES);

    // Cerrar todos los semaforos
    sem_destroy(&mutex);
    sem_destroy(&writer);
```

```
    return 0;
}
```