udp FACULTAD DE
INGENIERÍA Y CIENCIAS

Sistemas Operativos: Tarea 2

Diego Banda, Diego Salazar
Profesor: Victor Reyes
Ayudantes: Loreto Ñancucheo, Carlos Ruiz
Sección 2

3 de Noviembre, 2024

Índice

1. Introducción

En el presente informe se busca dar solución a un problema propuesto, utilizando *threads/hilos* y herramientas de sincronización. ¿Qué es un *thread*? Un *thread* es un subproceso de un proceso principal, utilizado para ejecutar 'paralelamente' una función o cálculo normalmente más simple que un trabajo para un proceso entero. Por otro lado, las herramientas de sincronización son utilizadas para evitar problemas en las ejecuciones de los *threads* cuando estos trabajan sobre una sección crítica (extracto de código, la cual trabaja sobre variables globales que pueden ser modificadas por cualquier subproceso), la modificación o uso de condicionales de estas variables globales puede hacer que se interfieran entre sí los procesos, llevando a una mala ejecución del código y que este no cumpla con su objetivo.

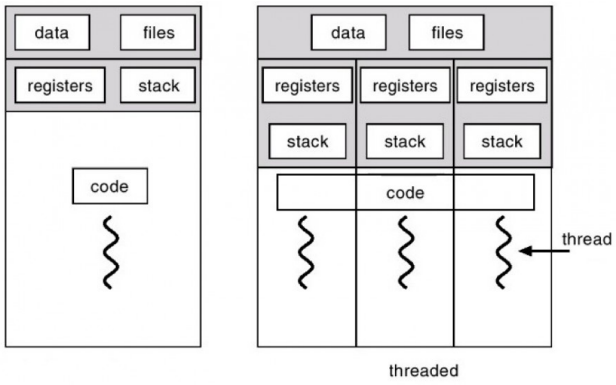

Figura 1: Ejemplo gráfico de proceso con threads

El enunciado del problema propuesto es el siguiente: Los profesores de la EIT están desbordados de trabajo corrigiendo sus pruebas y realizando sus otros quehaceres (investigación, gestión, atención de alumnos, etc.) relativos a la operación de la escuela. En los pocos momentos en los que no están trabajando, comentan sobre series que dan en servicios de streaming que han visto en sus horas fuera del trabajo. Como los precios de todo están yéndose a las nubes, ya todos los profes no tienen plata para poder tener todos los servicios, por lo que han llegado al consenso de repartirse las series y contárselas a los demás.

También se solicita (después del primer código desarrollado, en este mismo) realizar cambios para inducir un *deadlock* y otro para inducir un *livelock* en la ejecución del programa, pero, ¿Qué son cada uno? En ambos casos ocurren por técnicas de sincronización mal ejecutadas.

- **Deadlock:** Durante la ejecución, dos o más *threads* se quedan bloqueados esperando que el/los otros liberen un recurso (normalmente elementos de sincronización) para seguir con su ejecución, ninguno puede avanzar porque se bloquean mutuamente.
- **Livelock:** Para este caso no se bloquean, se interfieren entre sí, causando que efectivamente, los *threads* estén trabajando y avanzando en sus códigos pero sin salir de un ciclo, generando actividad pero sin progreso real.

2. Desarrollo

2.1. Parte 1

Para el desarrollo de la solución primero se deben tener en cuenta los siguientes puntos, los cuales son cruciales para el buen funcionamiento y camino que debe seguir el código.

- Hay 6 profesores que usan la plataforma Dasney y otros 6 profesores que usan la plataforma Betflix.
- Cada plataforma saca entre 10 y 15 random por semana y por plataforma.
- Cada profesor ve entre 0.5 y 2 series a la semana (0.5, 1, 1.5, 2 - random por cada profesor).
- El programa debe ejecutarse para 1 mes, 6 meses y 1 año.
- Durante ejecución debe mostrar comportamiento de los profesores y las plataformas, y enseñar estadísticas de series vistas en cada periodo.

2.1.1. Variables Globales

Por lo tanto, para llegar a una solución, primero se deben entender las variables globales a utilizar, las cuales se dividen principalmente en dos secciones, las que son para el desarrollo y solución del problema (aquí se encuentran los *mutex* y *cond* utilizados para sincronización también), y por otro lado las que son para almacenar y posteriormente enseñar estadísticas de lo que está ocurriendo u ocurrió.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdbool.h>

//  Variables Globales
int semanas;
float posibleSeries[4] = {0.5, 1, 1.5, 2}; // Tiempos posibles
float seriesDasney = 0; //  Seies en Dasney
float seriesBetflix = 0;    //  Series en Betflix
bool cambioDasney = false;
bool cambioBetflix = false;
bool estPeriodo = false;
pthread_mutex_t mutexDasney;
pthread_mutex_t mutexBetflix;
pthread_mutex_t mutexBarrier;
pthread_cond_t barrier_cond;
int barrierCount = 0;
//  Para estadisticas
```

```
20 float seriesTotalesDasney = 0;
21 float seriesTotalesBetflix = 0;
22 float vistoPorProfesor[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
23 float seriesVistasPeriodo = 0;
```

Listing 1: Variables Globales e inicio de código

- **Desarrollo y ejecución**
  - **semanas:** Cantidad de semanas a calcular, cantidad definida a través de main.
  - **posibleSeries[4]:** Cantidades de series posibles que puede ver un profesor en una semana.
  - **seriesDasney y seriesBetflix:** Series aún sin ver por profesores, inician en 0 ya que ninguna plataforma ha subido series en un inicio y varían a través del tiempo según series subidas y series vistas.
  - **cambioDasney, cambioBetflix y estPeriodo:** Variables booleanas que determinan si cambios que se deben hacer solo 1 vez por iteración ya lo realizó un thread o no.
  - **mutexDasney y mutexBetflix:** Variables de sincronización para secciones criticas respecto a profesores que ven Dasney o que ven Betflix.
  - **mutexBarrier, barrier_cond y barrierCount:** Variables utilizadas para crear barrera.
- **Estadísticas**
  - **seriesTotalesDasney y seriesTotalesBetflix:** Cantidad total de series subidas a las plataformas.
  - **vistoPorProfesor[12]:** Cantidad de series vistas por cada profesor.
  - **seriesVistasPeriodo:** Cantidad de series vistas en cada periodo (semana), este se reinicia en cada iteración.

2.1.2. Funciones

Las siguientes funciones son pequeñas pero necesarias para la ejecución del problema, ya que nos entregan los cálculos random necesarios, hechos cada semana (o iteración).

```
1 float nuevasSeries(){
2     return ((rand()%6)+10);
3 }
4
5 float seriesVistas(){
6     return posibleSeries[rand()%4];
7 }
```

Listing 2: Funciones para generación random

- **nuevasSeries():** Retorna un número random entre 10 y 15, utilizada para calcular las series añadidas a cada plataforma cada semana.
- **seriesVistas():** Retorna un número random dentro del arreglo 'posiblesseries', el número generado con 'rand() %4' oscila entre 0 y 3, los cuales son los indices del arreglo.

La siguiente función 'my_pthread_barrier_wait()' realiza exactamente lo mismo que una *barrier*, a través de su contador definido en 0 y con tope máximo 12 (cantidad de *threads*/profesores) hace que todos los *threads* deban llegar hasta este llamado a función para seguir con la ejecución de la siguiente parte del código, basicamente hace lo mismo que un 'pthread_barrier_t'.

```
void my_pthread_barrier_wait(){
    pthread_mutex_lock(&mutexBarrier);
    barrierCount++;
    if(barrierCount < 12){
        pthread_cond_wait(&barrier_cond, &mutexBarrier);
    }
    else{
        barrierCount = 0;
        pthread_cond_broadcast(&barrier_cond);
    }
    pthread_mutex_unlock(&mutexBarrier);
}
```

Listing 3: barrier

2.1.3. 'Main'

El apartado 'main' realiza la iniciación de las distintas variables y funciones de sincronización, define la cantidad de semanas a calcular, crea los distintos *threads*, espera que terminen de ejecutarse y finalmente muestra estadísticas generales.

```
int main(){
    int a;
    pthread_t profesores[12];   //Id desde 0 a 5 Dasney, desde 6 a
       11 Betflix
    int threads_ids[12];
    pthread_mutex_init(&mutexDasney, NULL);
    pthread_mutex_init(&mutexBetflix, NULL);
    pthread_mutex_init(&mutexBarrier, NULL);
    pthread_cond_init(&barrier_cond, NULL);
```

Listing 4: 'Main' iniciación

En este extracto de 'main' se inicializan algunas variables para su posterior uso dentro de esta misma función y también se inicializan los *mutex* que se crean en la sección de *Variables Globales*.

- **int a:** Variable que se usa después para el ingreso de las opciones (semanas a calcular).
- **pthread_t profesores[12]:** Arreglo del tipo *pthread_t* para guardar los ids de los *threads* que se crearán.
- **threads_ids[12]:** para guardar los ids, esto se hace para que posteriormente, al llamar a la función hecha para los *threads* se le pueda pasar el id en forma de puntero para su correcto funcionamiento.

```
1   //  Para el ingreso de las distintas opciones
2   printf("Ingrese␣tiempo␣a␣calcular\n");
3   printf("1␣=␣1␣mes\n2␣=␣6␣meses\n3␣=␣1␣ano\n");
4   scanf("%d", &a);
5   switch (a)
6   {
7   case 1:
8       semanas = 4;
9       break;
10  case 2:
11      semanas = 4*6;
12      break;
13  case 3:
14      semanas = 4*12;
15      break;
16  default:
17      printf("Debe␣ingresar␣numero␣valido\n");
18      return 0;
19  }
```

Listing 5: 'Main' selección

Aquí se pide por teclado el ingreso de un número para las distintas opciones de cálculo, 1 para 1 mes, 2 para 6 meses y 3 para 1 año, en caso de que no se ingrese una de estas opciones, el código terminará su ejecución.

```
1   //  Inicio calculos
2   printf("Ejecucion...\n");
3   for(int i = 0 ; i < 12 ; i++){
4       threads_ids[i] = i;
5       pthread_create(&profesores[i], NULL, mirarSeries, &
            threads_ids[i]);
6   }
7   for(int i = 0 ; i < 12 ; i++){
8       pthread_join(profesores[i], NULL);
9   }
```

Listing 6: 'Main' ejecución cálculos

Se inicia el cálculo, el primer ciclo *for* crea los threads y les asigna la función, junto a entregarle el puntero del id. Posteriormente el siguiente ciclo itera sobre el arreglo de profesores, haciendo *pthread_join* sobre cada uno, la función *join* hace que la ejecución del programa principal espere el término de tal *thread*.

```
    //  Muestra de estadisticas
    seriesTotales = seriesTotalesBetflix+seriesTotalesDasney;
    seriesVistasEst = seriesTotales - (seriesDasney + seriesBetflix)
       ;
    printf("Estadisticas...\n");
    printf("Series␣Totales:␣%.1f\nSeries␣Vistas:␣%.1f\nSeries␣
       Faltantes:␣%.1f\n", seriesTotales, seriesVistasEst,
       seriesDasney+seriesBetflix);
    printf("Dasney\n");
    printf("Series␣en␣Dasney:␣%.1f\nSeries␣Vistas:␣%.1f\nSeries␣
       Faltantes:␣%.1f\n", seriesTotalesDasney, seriesTotalesDasney-
       seriesDasney, seriesDasney);
    printf("Betflix\n");
    printf("Series␣en␣Betflix:␣%.1f\nSeries␣Vistas:␣%.1f\nSeries␣
       Faltantes:␣%.1f\n", seriesTotalesBetflix,
       seriesTotalesBetflix-seriesBetflix, seriesBetflix);
    printf("Minutos␣vistos␣por␣profesor:\n");
    for(int i = 0 ; i < 12 ; i++){
        printf("profesor␣%d␣ha␣visto␣%.1f␣series.\n", i+1,
           vistoPorProfesor[i]);
    }

    pthread_mutex_destroy(&mutexDasney);
    pthread_mutex_destroy(&mutexBetflix);
    pthread_mutex_destroy(&mutexBarrier);
    pthread_cond_destroy(&barrier_cond);
}
```

Listing 7: 'Main' estadísticas

Finalmente se calculan algunas estadísticas como *seriesTotales* o *seriesVistasEst* y mostrarlas por pantalla, también las vistas según plataforma y vistas por profesores, antes de la finalización, se terminan los *mutex* utilizados.

2.1.4. Función *threads*

Y esta es la función que ejecutan los *threads*, y en la cual hay que tener cuidado con las variables globales en las respectivas secciones críticas.

```
void* mirarSeries(void* arg){
    int id = *(int*)arg;
    for(int i = 0 ; i < semanas ; i++){
```

```
4           //  Esperar a todos los threads para iniciar una nueva
               semana
5           my_pthread_barrier_wait();
6           seriesVistasPeriodo = 0;
7           estPeriodo = false;
8           my_pthread_barrier_wait();
9           if(id < 6){ //  Dasney
10              //  Proceso de agregar nuevas series
11              pthread_mutex_lock(&mutexDasney);
12              //  Variables cambio indica si ya se subieron las nuevas
                    series o no
13              if(!cambioDasney){
14                  float nuevas = nuevasSeries();
15                  seriesDasney += nuevas;
16                  seriesTotalesDasney += nuevas;
17                  printf("Semana␣%d,␣", i+1);
18                  printf("en␣Dasney␣se␣subieron␣%.1f␣series,␣total:␣
                        %.1f,␣no␣vistas:␣%.1f\n", nuevas,
                        seriesTotalesDasney, seriesDasney);
19                  cambioDasney = true;
20              }
21              pthread_mutex_unlock(&mutexDasney);
22              //  Proceso de mirar serie
23              float visto = seriesVistas();
24              pthread_mutex_lock(&mutexDasney);
25              if(visto < seriesDasney){
26                  seriesDasney -= visto;
27                  vistoPorProfesor[id] += visto;
28              }else{
29                  visto = seriesDasney;
30                  seriesDasney = 0;
31                  vistoPorProfesor[id] += visto;
32              }
33              seriesVistasPeriodo += visto;
34              printf("Profesor␣de␣id␣%d␣vio␣%.1f␣series\n", id, visto)
                    ;
35              pthread_mutex_unlock(&mutexDasney);
36          }else{  //  Betflix
37              pthread_mutex_lock(&mutexBetflix);
38              if(!cambioBetflix){
39                  float nuevas = nuevasSeries();
40                  seriesBetflix += nuevas;
41                  seriesTotalesBetflix += nuevas;
42                  printf("Semana␣%d,␣", i+1);
43                  printf("en␣Betflix␣se␣subieron␣%.1f␣series,␣total␣
                        %.1f,␣no␣vistas:␣%.1f\n", nuevas,
```

```
                    seriesTotalesBetflix, seriesBetflix);
44                  cambioBetflix = true;
45              }
46              pthread_mutex_unlock(&mutexBetflix);
47              //  Proceso de mirar serie
48              float visto = seriesVistas();
49              pthread_mutex_lock(&mutexBetflix);
50              if(visto < seriesBetflix){
51                  seriesBetflix -= visto;
52                  vistoPorProfesor[id] += visto;
53              }else{
54                  visto = seriesBetflix;
55                  seriesBetflix = 0;
56                  vistoPorProfesor[id] += visto;
57              }
58              seriesVistasPeriodo += visto;
59              printf("Profesor de id %d vio %.1f series\n", id, visto)
                    ;
60              pthread_mutex_unlock(&mutexBetflix);
61          }
62          //  Espera que todos los threads terminen la semana
63          my_pthread_barrier_wait();
64          //  Estadistica periodo, solo 1 thread entra para mostrar
                solo 1 vez
65          pthread_mutex_lock(&mutexDasney);
66          pthread_mutex_lock(&mutexBetflix);
67          if(!estPeriodo){
68              printf("Series vistas en periodo (semana) %d: %.1f\n", i
                    +1, seriesVistasPeriodo);
69              estPeriodo = true;
70          }
71          pthread_mutex_unlock(&mutexBetflix);
72          pthread_mutex_unlock(&mutexDasney);
73          cambioDasney = false;
74          cambioBetflix = false;
75      }
76      return NULL;
77  }
```

Listing 8: Función para *threads*

Esta función ocurre en su mayoría dentro de un ciclo *for*, el cual va desde 0 hasta menor que las semanas elegidas inicialmente, dentro de este lo primero que encontramos es el llamado a la función *my_pthread_barrier_wait()* para que todos los threads inicien los ciclos a la vez, *seriesVistasPeriodo* y *estPeriodo* son para las estadísticas semanales a enseñar. El condicional *if* separa el código en dos principales códigos los cuales son prácticamente iguales,

la división es para las dos distintas plataformas, dentro de este condicional lo primero que pasa es que se bloquea una parte del código para que entre solo un *thread* con un *mutex*, dentro de este se verifica una variable, esto es para cambiar algunas variables globales solo una vez, el booleano se modifica y así los siguientes *threads* que vean esta instrucción no modificarán las variables, luego de este *mutex* y condicional se utiliza otro *mutex* para modificar las series vistas y que quedan por ver en cada plataforma, esto se divide según la cantidad de series por ver, también es bueno recordar que estos números son dados por funciones random. En cada periodo o semana, cada profesor enseñará lo visto, y luego del condicional que separa según plataforma, se enseñan estadísticas generales del periodo (esto se hace solo una vez, a través de *mutex* y la variable booleana *estPeriodo* solo un *thread* enseña las variables). Se sincroniza varias veces los términos en conjunto de los *threads* a través de *my_pthread_barrier_lock* para las estadísticas y distintos cambios en variables globales.

2.2. Parte 2

En esta sección se nos pide realizar cambios en el código con el fin de generar un **deadlock** y un **livelock**. Estos fueron creados en dos archivos de código distintos para mayor entendimiento y comprensión sobre la creación de errores que se nos pide.

2.2.1. Deadlock

Para la realización del **deadlock** se modifica la función *mirarSeries()* la cual es la encargada de la creación y manejo de los distintos threads que trabajan en el código, como se observa a continuación:

```
1  void* mirarSeries(void* arg){
2      int id = *(int*)arg;
3      for(int i = 0 ; i < semanas ; i++){
4          my_pthread_barrier_wait();
5          seriesVistasPeriodo = 0;
6          estPeriodo = false;
7          my_pthread_barrier_wait();
8
9          if(id < 6){ // Profesores de Dasney
10             pthread_mutex_lock(&mutexDasney);
11             if(!cambioDasney){
12                 float nuevas = nuevasSeries();
13                 seriesDasney += nuevas;
14                 seriesTotalesDasney += nuevas;
15                 printf("Semana␣%d,␣", i+1);
16                 printf("en␣Dasney␣se␣subieron␣%.1f␣series,␣total:␣
                       %.1f,␣no␣vistas:␣%.1f\n", nuevas,
                       seriesTotalesDasney, seriesDasney);
17                 cambioDasney = true;
18             }
```

```
19
20              float visto = seriesVistas();
21              pthread_mutex_lock(&mutexBetflix);
22              if(visto < seriesBetflix){
23                  seriesBetflix -= visto;
24                  vistoPorProfesor[id] += visto;
25              }else{
26                  visto = seriesBetflix;
27                  seriesBetflix = 0;
28                  vistoPorProfesor[id] += visto;
29              }
30              seriesVistasPeriodo += visto;
31              printf("Profesor␣de␣id␣%d␣vio␣%.1f␣series\n", id, visto)
                    ;
32              pthread_mutex_unlock(&mutexBetflix);
33              pthread_mutex_unlock(&mutexDasney);
34          } else { // Profesores de Betflix
35              pthread_mutex_lock(&mutexBetflix);
36              if(!cambioBetflix){
37                  float nuevas = nuevasSeries();
38                  seriesBetflix += nuevas;
39                  seriesTotalesBetflix += nuevas;
40                  printf("Semana␣%d,␣", i+1);
41                  printf("en␣Betflix␣se␣subieron␣%.1f␣series,␣total␣
                        %.1f,␣no␣vistas:␣%.1f\n", nuevas,
                        seriesTotalesBetflix, seriesBetflix);
42                  cambioBetflix = true;
43              }
44
45              float visto = seriesVistas();
46              pthread_mutex_lock(&mutexDasney);
47              if(visto < seriesDasney){
48                  seriesDasney -= visto;
49                  vistoPorProfesor[id] += visto;
50              }else{
51                  visto = seriesDasney;
52                  seriesDasney = 0;
53                  vistoPorProfesor[id] += visto;
54              }
55              seriesVistasPeriodo += visto;
56              printf("Profesor␣de␣id␣%d␣vio␣%.1f␣series\n", id, visto)
                    ;
57              pthread_mutex_unlock(&mutexDasney);
58              pthread_mutex_unlock(&mutexBetflix);
59          }
60          my_pthread_barrier_wait();
```

```
61
62          pthread_mutex_lock(&mutexDasney);
63          pthread_mutex_lock(&mutexBetflix);
64          if(!estPeriodo){
65              printf("Series vistas en periodo (semana) %d: %.1f\n", i
                    +1, seriesVistasPeriodo);
66              estPeriodo = true;
67          }
68          pthread_mutex_unlock(&mutexBetflix);
69          pthread_mutex_unlock(&mutexDasney);
70
71          cambioDasney = false;
72          cambioBetflix = false;
73      }
74      return NULL;
75  }
```

Listing 9: Creacion deadlock

2.2.2. Livelock

Para producir un **livelock** se modifica nuevamente la función encargada de la creación y funcionamiento de los threads. Obteniendo lo siguiente:

```
1   void* mirarSeries(void* arg){
2       int id = *(int*)arg;
3       for(int i = 0 ; i < semanas ; i++){
4           my_pthread_barrier_wait();
5           seriesVistasPeriodo = 0;
6           estPeriodo = false;
7           my_pthread_barrier_wait();
8
9           // Iniciar un intento de acceder a Dasney o Betflix
10          bool tryingDasney = (id < 6);  // true si el profesor esta
                en Dasney
11
12          while (1) {
13              if (tryingDasney) {
14                  if (pthread_mutex_trylock(&mutexDasney) == 0) {
15                      // Proceso de agregar nuevas series
16                      if (!cambioDasney) {
17                          float nuevas = nuevasSeries();
18                          seriesDasney += nuevas;
19                          seriesTotalesDasney += nuevas;
20                          printf("Semana %d, ", i + 1);
```

```
21                          printf("en␣Dasney␣se␣subieron␣%.1f␣series,␣
                               total:␣%.1f,␣no␣vistas:␣%.1f\n", nuevas,
                               seriesTotalesDasney, seriesDasney);
22                          cambioDasney = true;
23                      }
24
25                      // Proceso de mirar serie
26                      float visto = seriesVistas();
27                      if (visto < seriesDasney) {
28                          seriesDasney -= visto;
29                          vistoPorProfesor[id] += visto;
30                      } else {
31                          visto = seriesDasney;
32                          seriesDasney = 0;
33                          vistoPorProfesor[id] += visto;
34                      }
35                      seriesVistasPeriodo += visto;
36                      printf("Profesor␣de␣id␣%d␣vio␣%.1f␣series\n", id
                           , visto);
37                      pthread_mutex_unlock(&mutexDasney);
38                      break;
39                  } else {
40                      printf("Profesor␣de␣id␣%d␣no␣pudo␣acceder␣a␣
                           Dasney,␣cediendo...\n", id);
41                      usleep(100000);
42                  }
43              } else { // Acceso a Betflix
44                  if (pthread_mutex_trylock(&mutexBetflix) == 0) {
45                      // Proceso de agregar nuevas series
46                      if (!cambioBetflix) {
47                          float nuevas = nuevasSeries();
48                          seriesBetflix += nuevas;
49                          seriesTotalesBetflix += nuevas;
50                          printf("Semana␣%d,␣", i + 1);
51                          printf("en␣Betflix␣se␣subieron␣%.1f␣series,␣
                               total␣%.1f,␣no␣vistas:␣%.1f\n", nuevas,
                               seriesTotalesBetflix, seriesBetflix);
52                          cambioBetflix = true;
53                      }
54
55                      // Proceso de mirar serie
56                      float visto = seriesVistas();
57                      if (visto < seriesBetflix) {
58                          seriesBetflix -= visto;
59                          vistoPorProfesor[id] += visto;
60                      } else {
```

```
61                          visto = seriesBetflix;
62                          seriesBetflix = 0;
63                          vistoPorProfesor[id] += visto;
64                      }
65                      seriesVistasPeriodo += visto;
66                      printf("Profesor de id %d vio %.1f series\n", id
                            , visto);
67                      pthread_mutex_unlock(&mutexBetflix);
68                      break;
69                  } else {
70
71                      printf("Profesor de id %d no pudo acceder a
                            Betflix, cediendo...\n", id);
72                      usleep(100000);  // Espera antes de volver a
                            intentar
73                  }
74              }
75          }
76          my_pthread_barrier_wait();
77
78          pthread_mutex_lock(&mutexDasney);
79          pthread_mutex_lock(&mutexBetflix);
80          if (!estPeriodo) {
81              printf("Series vistas en periodo (semana) %d: %.1f\n", i
                    + 1, seriesVistasPeriodo);
82              estPeriodo = true;
83          }
84          pthread_mutex_unlock(&mutexBetflix);
85          pthread_mutex_unlock(&mutexDasney);
86          cambioDasney = false;
87          cambioBetflix = false;
88      }
89      return NULL;
90  }
```

Listing 10: Creacion Livelock

3. Resultados

3.1. Parte 1

Como se solicita el cálculo para distintas fechas, cuando el código es ejecutado se pide por consola un número para determinar cuanto tiempo es el que quieres calcular. Se debe considerar que por la cantidad de iteraciones y *prints* que hay en la ejecución no es posible enseñar todo en una sola foto.

3.1.1. Ejecución 1 mes

```
(base) diego@MacBook-Air-de-Diego Tarea2 % ./parte1
Ingrese tiempo a calcular
1 = 1 mes
2 = 6 meses
3 = 1 ano
1
Ejecucion...
Semana 1, en Dasney se subieron 11.0 series, total: 11.0, no vistas: 11.0
Profesor de id 1 vio 1.0 series
Semana 1, en Betflix se subieron 11.0 series, total 11.0, no vistas: 11.0
Profesor de id 8 vio 1.5 series
Profesor de id 4 vio 1.5 series
Profesor de id 9 vio 0.5 series
Profesor de id 3 vio 0.5 series
Profesor de id 6 vio 1.5 series
Profesor de id 0 vio 2.0 series
Profesor de id 10 vio 1.0 series
Profesor de id 5 vio 0.5 series
Profesor de id 2 vio 1.0 series
Profesor de id 7 vio 0.5 series
Profesor de id 11 vio 1.5 series
Series vistas en periodo (semana) 1: 13.0
Semana 2, en Betflix se subieron 13.0 series, total 24.0, no vistas: 17.5
Profesor de id 9 vio 2.0 series
Profesor de id 6 vio 0.5 series
Profesor de id 7 vio 0.5 series
Semana 2, en Dasney se subieron 15.0 series, total: 26.0, no vistas: 19.5
Profesor de id 1 vio 2.0 series
Profesor de id 8 vio 1.0 series
Profesor de id 11 vio 1.0 series
Profesor de id 10 vio 1.0 series
Profesor de id 4 vio 1.0 series
Profesor de id 5 vio 0.5 series
Profesor de id 3 vio 1.0 series
Profesor de id 2 vio 2.0 series
Profesor de id 0 vio 1.5 series
Series vistas en periodo (semana) 2: 14.0
Semana 3, en Dasney se subieron 12.0 series, total: 38.0, no vistas: 23.5
Profesor de id 2 vio 1.0 series
Profesor de id 1 vio 1.5 series
Profesor de id 4 vio 0.5 series
Profesor de id 3 vio 2.0 series
Semana 3, en Betflix se subieron 11.0 series, total 35.0, no vistas: 22.5
Profesor de id 9 vio 1.0 series
Profesor de id 6 vio 2.0 series
Profesor de id 8 vio 1.0 series
Profesor de id 11 vio 2.0 series
Profesor de id 10 vio 1.0 series
Profesor de id 7 vio 2.0 series
Profesor de id 0 vio 1.5 series
Profesor de id 5 vio 0.5 series
Series vistas en periodo (semana) 3: 16.0
Semana 4, en Dasney se subieron 11.0 series, total: 49.0, no vistas: 27.5
```

Figura 2: Ejecución 1 mes - imagen 1

```
Semana 4, en Dasney se subieron 11.0 series, total: 49.0, no vistas: 27.5
Profesor de id 0 vio 0.5 series
Profesor de id 1 vio 1.0 series
Profesor de id 3 vio 1.0 series
Profesor de id 2 vio 2.0 series
Profesor de id 4 vio 1.5 series
Profesor de id 5 vio 1.0 series
Semana 4, en Betflix se subieron 12.0 series, total 47.0, no vistas: 25.5
Profesor de id 9 vio 1.0 series
Profesor de id 6 vio 1.0 series
Profesor de id 10 vio 0.5 series
Profesor de id 7 vio 0.5 series
Profesor de id 8 vio 0.5 series
Profesor de id 11 vio 1.5 series
Series vistas en periodo (semana) 4: 12.0
Estadisticas...
Series Totales: 96.0
Series Vistas: 55.0
Series Faltantes: 41.0
Dasney
Series en Dasney: 49.0
Series Vistas: 28.5
Series Faltantes: 20.5
Betflix
Series en Betflix: 47.0
Series Vistas: 26.5
Series Faltantes: 20.5
Minutos vistos por profesor:
profesor 1 ha visto 5.5 series.
profesor 2 ha visto 5.5 series.
profesor 3 ha visto 6.0 series.
profesor 4 ha visto 4.5 series.
profesor 5 ha visto 4.5 series.
profesor 6 ha visto 2.5 series.
profesor 7 ha visto 5.0 series.
profesor 8 ha visto 3.5 series.
profesor 9 ha visto 4.0 series.
profesor 10 ha visto 4.5 series.
profesor 11 ha visto 3.5 series.
profesor 12 ha visto 6.0 series.
(base) diego@MacBook-Air-de-Diego Tarea2 %
```

Figura 3: Ejecución 1 mes - imagen 2

3.1.2. Ejecución 6 meses

A partir de los 6 meses, los cálculos que se enseñan por pantalla son muchos, por lo tanto se enseñará solo el final del código con las estadísticas finales.

```
Series vistas en periodo (semana) 23: 16.5
Semana 24, en Betflix se subieron 15.0 series, total 305.0, no vistas: 126.5
Profesor de id 7 vio 0.5 series
Profesor de id 9 vio 1.0 series
Profesor de id 6 vio 1.0 series
Profesor de id 8 vio 2.0 series
Profesor de id 10 vio 1.5 series
Semana 24, en Dasney se subieron 15.0 series, total: 306.0, no vistas: 147.0
Profesor de id 11 vio 0.5 series
Profesor de id 2 vio 1.0 series
Profesor de id 1 vio 1.0 series
Profesor de id 3 vio 1.0 series
Profesor de id 4 vio 0.5 series
Profesor de id 5 vio 1.5 series
Profesor de id 0 vio 0.5 series
Series vistas en periodo (semana) 24: 12.0
Estadisticas...
Series Totales: 611.0
Series Vistas: 349.5
Series Faltantes: 261.5
Dasney
Series en Dasney: 306.0
Series Vistas: 164.5
Series Faltantes: 141.5
Betflix
Series en Betflix: 305.0
Series Vistas: 185.0
Series Faltantes: 120.0
Minutos vistos por profesor:
profesor 1 ha visto 28.5 series.
profesor 2 ha visto 29.0 series.
profesor 3 ha visto 23.5 series.
profesor 4 ha visto 27.0 series.
profesor 5 ha visto 29.5 series.
profesor 6 ha visto 27.0 series.
profesor 7 ha visto 28.0 series.
profesor 8 ha visto 31.5 series.
profesor 9 ha visto 31.0 series.
profesor 10 ha visto 29.5 series.
profesor 11 ha visto 34.5 series.
profesor 12 ha visto 30.5 series.
(base) diego@MacBook-Air-de-Diego Tarea2 %
```

Figura 4: Ejecución 6 meses

3.1.3. Ejecución 1 año

```
Series vistas en periodo (semana) 47: 18.0
Semana 48, en Betflix se subieron 15.0 series, total 599.0, no vistas: 253.5
Profesor de id 6 vio 1.5 series
Profesor de id 10 vio 1.5 series
Profesor de id 8 vio 2.0 series
Profesor de id 11 vio 0.5 series
Profesor de id 9 vio 0.5 series
Profesor de id 7 vio 1.0 series
Semana 48, en Dasney se subieron 14.0 series, total: 597.0, no vistas: 246.5
Profesor de id 5 vio 1.5 series
Profesor de id 1 vio 1.0 series
Profesor de id 4 vio 1.0 series
Profesor de id 0 vio 0.5 series
Profesor de id 3 vio 1.0 series
Profesor de id 2 vio 1.0 series
Series vistas en periodo (semana) 48: 13.0
Estadisticas...
Series Totales: 1196.0
Series Vistas: 709.0
Series Faltantes: 487.0
Dasney
Series en Dasney: 597.0
Series Vistas: 356.5
Series Faltantes: 240.5
Betflix
Series en Betflix: 599.0
Series Vistas: 352.5
Series Faltantes: 246.5
Minutos vistos por profesor:
profesor 1 ha visto 54.0 series.
profesor 2 ha visto 58.0 series.
profesor 3 ha visto 66.0 series.
profesor 4 ha visto 60.0 series.
profesor 5 ha visto 59.5 series.
profesor 6 ha visto 59.0 series.
profesor 7 ha visto 50.0 series.
profesor 8 ha visto 56.5 series.
profesor 9 ha visto 62.0 series.
profesor 10 ha visto 60.0 series.
profesor 11 ha visto 64.5 series.
profesor 12 ha visto 59.5 series.
(base) diego@MacBook-Air-de-Diego Tarea2 %
```

Figura 5: Ejecución 1 año

3.2. Parte 2

3.2.1. Deadlock

El resultado obtenido por la terminal al compilar y ejecutar el codigo con deadlock es el siguiente:

```
6semestre/SISTEMAS OPERATIVOS/TAREA2 via   v13.3.1-gcc
❯ gcc -pthread -o deadlock deadlock.c

6semestre/SISTEMAS OPERATIVOS/TAREA2 via   v13.3.1-gcc
❯ ./deadlock
Ingrese tiempo a calcular
1 = 1 mes
2 = 6 meses
3 = 1 ano
2
Ejecucion...
Semana 1, en Dasney se subieron 14.0 series, total: 14.0, no vistas: 14.0
Semana 1, en Betflix se subieron 11.0 series, total 11.0, no vistas: 11.0
```

Figura 6: Comprobación deadlock

Como se observa en la figura se nota que el código se esta ejecutando pero sin avanzar de forma que se comprueba el resultado esperado, obtener un deadlock.

3.2.2. Livelock

El resultado obtenido por la terminal al compilar y ejecutar el código con livelock es el siguiente:

```
Profesor de id 2 vio 0.5 series
Series vistas en periodo (semana) 21: 15.0
Semana 22, en Betflix se subieron 12.0 series, total 275.0, no vistas: 115.
Profesor de id 10 vio 1.5 series
Profesor de id 11 no pudo acceder a Betflix, cediendo...
Semana 22, Profesor de id 8 no pudo acceder a Betflix, cediendo...
Profesor de id 4 no pudo acceder a Dasney, cediendo...
Profesor de id 3 no pudo acceder a Dasney, cediendo...
Profesor de id 6 no pudo acceder a Betflix, cediendo...
Profesor de id 7 vio 1.5 series
Profesor de id 9 no pudo acceder a Betflix, cediendo...
Profesor de id 1 no pudo acceder a Dasney, cediendo...
Profesor de id 5 no pudo acceder a Dasney, cediendo...
en Dasney se subieron 11.0 series, total: 270.0, no vistas: 118.5
Profesor de id 0 vio 0.5 series
Profesor de id 2 no pudo acceder a Dasney, cediendo...
Profesor de id 11 vio 0.5 series
Profesor de id 3 no pudo acceder a Dasney, cediendo...
Profesor de id 8 no pudo acceder a Betflix, cediendo...
Profesor de id 6 vio 0.5 series
Profesor de id 4 vio 1.5 series
Profesor de id 9 vio 0.5 series
Profesor de id 1 vio 1.5 series
Profesor de id 5 no pudo acceder a Dasney, cediendo...
```

Figura 7: Comprobación livelock parte 1

```
Profesor de id 2 vio 1.0 series
Profesor de id 7 vio 1.0 series
Profesor de id 1 vio 1.0 series
Profesor de id 6 no pudo acceder a Betflix, cediendo...
Profesor de id 10 vio 1.5 series
Profesor de id 3 vio 1.5 series
Profesor de id 6 vio 2.0 series
Series vistas en periodo (semana) 43: 15.0
Semana 44, en Dasney se subieron 11.0 series, total: 530.0, no vistas: 216.
Profesor de id 3 vio 1.0 series
Profesor de id 6 no pudo acceder a Betflix, cediendo...
Profesor de id 0 no pudo acceder a Dasney, cediendo...
Profesor de id 7 no pudo acceder a Betflix, cediendo...
Profesor de id 4 no pudo acceder a Dasney, cediendo...
Profesor de id 9 no pudo acceder a Betflix, cediendo...
Profesor de id 10 no pudo acceder a Betflix, cediendo...
Profesor de id 2 vio 2.0 series
Profesor de id 5 no pudo acceder a Dasney, cediendo...
Profesor de id 11 no pudo acceder a Betflix, cediendo...
Semana 44, en Betflix se subieron 11.0 series, total 559.0, no vistas: 246.
Profesor de id 8 vio 1.5 series
Profesor de id 1 no pudo acceder a Dasney, cediendo...
Profesor de id 10 no pudo acceder a Betflix, cediendo...
Profesor de id 5 no pudo acceder a Dasney, cediendo...
```

Figura 8: Comprobación livelock parte 2

Como se observa en las dos imágenes anteriores el código se ejecuta constantemente mientras los threads esperan e intentan acceder a los mutex.

4. Análisis

4.1. Parte 1

Como se puede apreciar a través de un buen análisis de código y/o la ejecución de estos mismos, lo desarrollado cumple satisfactoriamente con calcular las series vistas utilizando *threads*, sin fallos de sincronización o inconsistencias en variables globales. Los resultados a través de distintas ejecuciones del código siempre serán distintas, por las distintas formas de avanzar a través del código de los hilos y los números random en los que se ve involucrado cada uno.

4.2. Parte 2

4.2.1. Deadlock

Para que ocurra el **deadlock** como se menciono anteriormente fue modificar la **función threads** de manera que los profesores que ven dasney, primero bloquean el **mutexDasney** y luego intentan bloquear el **mutexBetflix**. Por el contrario los profesores que ven Betflix primero bloquean el **mutexBetflix** para luego intentar bloquear el **mutexDasney** de esta forma se genera una espera circular donde los threads están bloqueados sin posibilidad de avanzar.

4.2.2. Livelock

Para que ocurra el **livelock** como se menciono anteriormente fue modificar la **función threads** ya que es la encargada del manejo y uso de estos. Se creo una variable booleana donde controla el acceso a Dasney o Betflix, recibiendo *TRUE* si el profesor accede a Dasney o *FALSE* para acceder a Betflix. El livelock ocurre ya que todo esto transcurre dentro de un bucle y usando la función $pthread_mutex_trylock$ en la cual se intenta bloquear un mutex, si no se consigue entra en un tiempo de espera, con el uso de *usleep*, antes de volver a intentar. Todo esto ocurre simultáneamente de forma alternada en la cual se produce un circulo de intentar y fallar.

5. Conclusión

Se aprendió la buena manera de analizar los diferentes problema que conlleva trabajar con *threads* y variables globales, es importante llevar una sincronización acorde al avance de estos por el código y el contexto del problema a resolver. El uso de distintas herramientas de sincronización ayuda a comprender el uso de estas y lo importantes que son para no irrumpir en el buen comportamiento del código y también no llevar al código en transformarse en uno más ineficiente que si se hiciera totalmente lineal.

Por otro lado, el provocar y crear código con los errores *deadlock* y *livelock* ayuda a la comprensión de estos errores, qué es lo que los provoca y de qué manera se identifican y a la vez se evitan, esto es importante para evitar errores a futuro.

Finalmente, se dan por comprendidos los temas de *threads* y sincronización, los cuales son de vital importancia para la eficiencia y buen comportamiento de los sistemas y códigos.