Programación Concurrente, Mayo 2023

Práctica 5

Martínez Buenrostro Jorge Rafael.
Universidad Autónoma Metropolitana
Unidad Iztapalapa, México
molap96@gmail.com

Actividad B

En esta actividad se tomó como base el ejemplo *EjempliMinRenos* visto en clase, en el que vimos el funcionamiento de las barreras. Se realizaron las siguientes modificaciones:

1. Se define el tamaño de la matriz como N, así como dos funciones: CalcularMaximo y CalcularMinimo las cuales serán utilizadas por los hilos para poder encontrar el valor máximo de cada fila y el mínimo de la lista de valores máximos. También se declara tanto la matriz como el arreglo en el que se guardará el valor máximo de cada fila. Y para terminar se declara la barrera que nos permitirá esperar a saber el valor máximo de todas las filas

```c
#define N 6

void CalcularMaximo(void *ptr);
void CalcularMinimo(void *ptr);
void LlenarMatriz();
void MostrarMatriz();
int matriz[N][N];
int resultados[N];

pthread_barrier_t EntregarMaximos;
```

2. Se crean dos funciones: la primera nos servirá para poder asignarle valores a cada casilla de la matriz y la segunda para poder mostrar en pantalla la misma.

```c
void LlenarMatriz(){
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            matriz[i][j]=rand()%50;
        }
    }
    printf("\n++++Matriz llena++++\n\n");
    MostrarMatriz();
}
```

```c
void MostrarMatriz(){
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            printf(" %d\t",matriz[i][j]);
        }
        printf("\n");
    }
}
```

3. Dentro de la función *CalcularMaximo* se usa un ciclo for en el que utiliza el id del hilo para poder asignarle una fila para realizar el cálculo, y guardar el resultado dentro del arreglo usando el mismo id como índice. Al final el hilo por medio de la barrera espera a que los demás hilos terminen.

```
void CalcularMaximo(void *ptr){
    int id,i,maximo;
    id=(intptr_t)ptr;
    maximo=0;
    for(i=0;i<N;i++){
        if(matriz[id][i]>maximo)
            maximo=matriz[id][i];
    }
    printf("El máximo del renglón %d es %d\n",id, maximo);
    fflush(stdout);
    resultados[id]=maximo;
    pthread_barrier_wait(&EntregarMaximos);
    pthread_exit(0);
}
```

4. Para terminar dentro de la función *CalcularMinimo* se busca dentro del arreglo de resultados el valor más pequeño para poder ponero en pantalla. Al final el hilo espera que todos terminen.

```
void CalcularMinimo(void *ptr){
    int i, minimo;
    minimo=51;
    printf("Maximos de cada renglon\n"); fflush(stdout);
    for(i=0;i<N;i++){
        printf("|%d|\n",resultados[i]); fflush(stdout);
    }
    for(i=0;i<N;i++){
        if(resultados[i]<minimo)
            minimo=resultados[i];
    }
    printf("El mínimo de los máximos es %d\n",minimo);
    pthread_barrier_wait(&EntregarMaximos);
    pthread_exit(0);
}
```

En cuanto a la inicialización, manejo y destrucción de los hilos y la barrera se queda prácticamente intacto.

Enlace GDB - https://onlinegdb.com/7UuCAm3wgA

Código Fuente

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

```c
#define N 6

void CalcularMaximo(void *ptr);
void CalcularMinimo(void *ptr);
void LlenarMatriz();
void MostrarMatriz();
int matriz[N][N];
int resultados[N];

pthread_barrier_t EntregarMaximos;

int main(int argc, char const *argv[])
{
   printf("\n***** Ejercicio B *****\n");
   printf("\n++++Creando y llenando matriz++++");
   int id[N],i;
   pthread_t Renglones[N],Minimo;
   srand(time(NULL));
   LlenarMatriz();
   pthread_barrier_init(&EntregarMaximos,NULL,N+1);

   for(i=0;i<N;i++){
       id[i]=i;

pthread_create(&Renglones[i],NULL,(void*)&CalcularMaximo,(void*)(intptr_t)id[i]);
   }
   pthread_create(&Minimo,NULL,(void*)&CalcularMinimo,NULL);

   pthread_join(Minimo,NULL);
   for(i=0;i<N;i++){
       pthread_join(Renglones[i],NULL);
   }
   pthread_barrier_destroy(&EntregarMaximos);
   return 0;
}

void CalcularMaximo(void *ptr){
```

```
    int id,i,maximo;
    id=(intptr_t)ptr;
    maximo=0;
    for(i=0;i<N;i++){
        if(matriz[id][i]>maximo)
            maximo=matriz[id][i];
    }
    printf("El máximo del renglón %d es %d\n",id, maximo);
    fflush(stdout);
    resultados[id]=maximo;
    pthread_barrier_wait(&EntregarMaximos);
    pthread_exit(0);
}

void CalcularMinimo(void *ptr){
    int i, minimo;
    minimo=51;
    printf("Maximos de cada renglon\n"); fflush(stdout);
    for(i=0;i<N;i++){
        printf("|%d|\n",resultados[i]); fflush(stdout);
    }
    for(i=0;i<N;i++){
        if(resultados[i]<minimo)
            minimo=resultados[i];
    }
    printf("El mínimo de los máximos es %d\n",minimo);
    pthread_barrier_wait(&EntregarMaximos);
    pthread_exit(0);
}

void LlenarMatriz(){
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            matriz[i][j]=rand()%50;
        }
    }
```

```
    printf("\n++++Matriz llena++++\n\n");
    MostrarMatriz();
}
void MostrarMatriz(){
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            printf(" %d\t",matriz[i][j]);
        }
        printf("\n");
    }
}
```

Actividad C

Para está actividad se tomó como base el ejercicio pasado. Al que únicamente se le hicieron las siguientes modificaciones:

1. La declaración de la constante ***NUMMAT*** que representa las veces que se repetirá el proceso visto en la actividad anterior.

```
#define NUMMAT 3
```

2. La declaración e inicialización de una nueva barrera llamada **"MostrarResultados"** la cuál espera a que se termine de mostrar en la terminal el mínimo de la lista de máximos.

3. Dentro de la función ***CalcularMaximo*** se agrega un ciclo que se repite ***NUMMAT*** veces. Dentro del ciclo se deja la funcionalidad de la función, la única diferencia es que al final se agrega una nueva espera para la nueva barrera.

```
for(i=0;i<NUMMAT;i++){
        maximo=0;
    for(j=0;j<N;j++){
        if(matriz[id][j]>maximo)
            maximo=matriz[id][j];
    }
    resultados[id]=maximo;
    pthread_barrier_wait(&MaximosListos);
    pthread_barrier_wait(&MostrarResultados);
}
```

4. Dentro de la función ***CalcularMinimo*** se agrega un ciclo que se repite ***NUMMAT*** veces. Dentro del cuál se deja funcionalidad original de la función, el primer cambio es que la primer línea dentro del ciclo es una espera para la barrera ***MaximosListos***, la cual se encarga de esperar a que se tengan los máximos de todos los renglones de la matriz, el resto del código dentro del ciclo se mantiene igual.

```
for(i=0;i<NUMMAT;i++){
    pthread_barrier_wait(&MaximosListos);
        minimo=51;

    printf("\nMaximos de cada renglon\n"); fflush(stdout);
    for(j=0;j<N;j++){
        printf("\t|%d|\n",resultados[j]); fflush(stdout);
    }
    for(j=0;j<N;j++){
        if(resultados[j]<minimo)
            minimo=resultados[j];
    }
    printf("\n  El mínimo de los máximos es %d\n",minimo);
    if(i<NUMMAT-1)
        LlenarMatriz();
    pthread_barrier_wait(&MostrarResultados);
}
```

Enlace GDB - https://onlinegdb.com/BMA1s-hnK

Código Fuente

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#define N 6
#define NUMMAT 5

void CalcularMaximo(void *ptr);
void CalcularMinimo(void *ptr);
void LlenarMatriz();
void MostrarMatriz();

pthread_barrier_t MaximosListos, MostrarResultados;
```

```
int matriz[N][N];
int resultados[N];

int main(int argc, char const *argv[]){
   int id[N],i;
   pthread_t Maximos[N],Minimo;

   pthread_barrier_init(&MaximosListos,NULL,N+1);
   pthread_barrier_init(&MostrarResultados,NULL,N+1);

   srand(time(NULL));
   LlenarMatriz();

   for(i=0;i<N;i++){
       id[i]=i;

pthread_create(&Maximos[i],NULL,(void*)&CalcularMaximo,(void*)(intptr_t)id[i]);
   }
   pthread_create(&Minimo,NULL,(void*)&CalcularMinimo,NULL);

   pthread_join(Minimo,NULL);
   for(i=0;i<N;i++){
       pthread_join(Maximos[i],NULL);
   }
   pthread_barrier_destroy(&MaximosListos);
   pthread_barrier_destroy(&MostrarResultados);
   return 0;
}

void CalcularMaximo(void *ptr){
   int id,i,j,maximo;
   id=(intptr_t)ptr;

   for(i=0;i<NUMMAT;i++){
           maximo=0;
       for(j=0;j<N;j++){
           if(matriz[id][j]>maximo)
```

```
                maximo=matriz[id][j];
        }
        resultados[id]=maximo;
        pthread_barrier_wait(&MaximosListos);
        pthread_barrier_wait(&MostrarResultados);
    }

    pthread_exit(0);
}

void CalcularMinimo(void *ptr){
    int i, j, minimo;

    for(i=0;i<NUMMAT;i++){
        pthread_barrier_wait(&MaximosListos);
            minimo=51;

        printf("\nMaximos de cada renglon\n"); fflush(stdout);
        for(j=0;j<N;j++){
            printf("\t|%d|\n",resultados[j]); fflush(stdout);
        }
        for(j=0;j<N;j++){
            if(resultados[j]<minimo)
                minimo=resultados[j];
        }
        printf("\n  El mínimo de los máximos es %d\n",minimo);
        if(i<NUMMAT-1)
            LlenarMatriz();
        pthread_barrier_wait(&MostrarResultados);
    }

    pthread_exit(0);
}

void LlenarMatriz(){
    int i,j;
    char car = '-';
```

```
   for(i=0;i<N;i++){
       for(j=0;j<N;j++){
           matriz[i][j]=rand()%50;
       }
   }
   printf("+--------------------------------------------------+");
   printf("\n\t\tMatriz nueva \n\n");
   MostrarMatriz();
}

void MostrarMatriz(){
   int i,j;
   for(i=0;i<N;i++){
       for(j=0;j<N;j++){
           printf(" %d\t",matriz[i][j]);
       }
       printf("\n");
   }
}
```