ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

FACULTAD DE INGENIERÍA Y CIENCIAS

$CIT2010$
Sistemas Operativos
Tarea 2: Threads y Sincronización

Pablo Moraga

2:pablo.moraga@mail.udp.cl

Profesor: Victor Reyes

Octubre 21, 2022

Índice

1. Introducción

2. Procedimiento

En esta sección se explicara paso a paso el código hecho. Estos son los valores con las descripciones que son uso de todas las funciones:

```
#define N 10  // Numero de threads
#define T 100 // para el semaforoS

using n
pthread                                    hreads
int mej                                    eso global
string caminoglobal;          // Camino del mejor peso
pthread_mutex_t lock;         // Inicializar variable de mutex
int num_sem = 0;
sem_t S[T];
```

std::string caminoglobal
Camino del mejor peso

Figura 1: Valores

2.1. Grafo

Para el hacer el grafo dirigido y acíclico, se hizo una estructura con nodos y otra estructura de aristas.
El nodo tiene los siguientes parámetros:

- **letra**: Este es un carácter CHAR.
- ***sgte**: Este es un puntero al siguiente nodo.
- ***ady**: Este es un puntero a su arista correspondiente.

La arista tiene los siguientes parámetros:

- **id**: Este es un id de la arista, su uso es para identificarlo con los semáforos.
- **peso**: Este es el peso de la arista.
- ***destino**: Hace referencia por puntero al nodo de destino(al que pertenece).
- ***sgte**: Es un puntero a la siguiente arista en el caso de tener este es una conexión del mismo nodo destino.

2.2. Funciones

Las funciones básicas para comenzar son:

- **insertar_nodo**: Esta es la principal función para comenzar la que hace los nodos, cuando se crea el primer nodo solo se agrega, pero para un nuevo nodo sera asignado a su puntero ***sgte** como el siguiente nodo, en caso de que el sgte nodo sea NULL este se agrega.

- **insertar_arista**:Este es para crear las conexiones de los nodos con sus respectivos pesos de camino. Se le pasa el nodo inicial, el nodo final, y su peso. También dentro de esta función se inicializan los **semáforos** a la id de la arista así se tiene a que semáforo sumar o restar valores para el bloqueo o paso a la arista. Al arista podrán entrar entre 1 a 5 randomizado threads.
- **agregar_arista**: Esta es llamada por la función anterior que hace los encuentros de los nodos y los pasa con el valor de la arista. Esta función hace la agregación de la arista.
- **mostrar_grafo**: Solo muestra el grafo.

2.3. Main

El primer paso es llamar a la función **inicializar_grafo** lo que hace es crear el grafo, el grafo que se crea es el de la figura 2, en el caso de querer cambiarlo es importante saber que se puede hacer cualquier tipo de grafo dirigido pero a tener en cuenta 2 condiciones, siempre empezara por el primer nodo a hacer en este caso el A y **siempre terminara en K** en este caso en la tarea debe llegar hasta KH pero como se utilizo la variable tipo char solo se puede una letra.

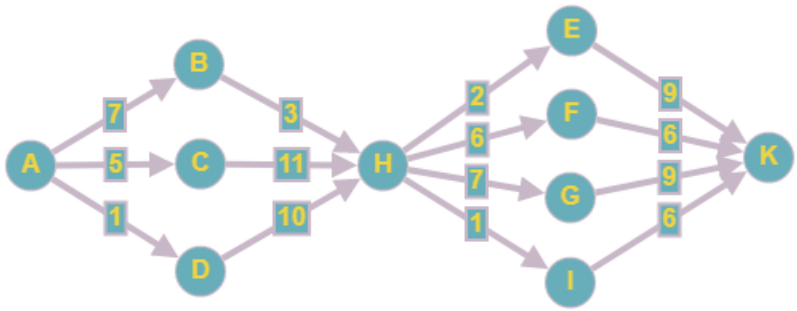

Figura 2: Grafo

Después se procede a mostrar el grafo, mas que nada se muestran las conexiones entre nodos y sus pesos para ir a tal nodo.
Por consiguiente se inicializa el mutex que se ocupara mas adelante y se crean los threads y se hace un join a su respectiva función de thread.

2.3.1. Función de Threads

Los threads tienen las siguientes variables:

- **mejorpeso**: Este guardara el mejor peso de **un solo thread** este no afecta en nada, es solo para saber el mejor peso que saca cada thread.
- **mejorcamino**: Este va de la mano con el anterior, pero en este caso guarda el camino.
- **pesototal**: esta variable va acumulando el peso mientras va de nodo en nodo.

- **camino**: Va de la mano con el anterior, va acumulando el camino cuando va de nodo en nodo.
- **nodo**: Es el nodo A(el primer nodo que se crea).
- **rep**: Cuantas veces el mismo thread recorrerá el grafo.

Se hace el recorrido rep veces y por cada vez los valores de pesototal, nodo y camino se inicializarán. Por consiguiente harán un ciclo hará llegar al último nodo en este caso K que representa KH de la tarea.
En el ciclo se hace un llamado a la función **cambionodo** que retorna el siguiente nodo y hace un cambio en el pesototal a través de un puntero.
Esta función mencionada hace un recorrido por todos los caminos que tiene el nodo actual y hace un conteo de estos, con este conteo se hace un random para ver por que camino irse, ya elegido el camino se activa el semáforo se sacan los pesos y se hace el recorrido para llegar al otro nodo y se finaliza el semáforo como se muestra en la figura 3.

```
sem_wait(&S[aris->id]);
int a = aris->peso;
*peso = *peso + a;
actual = aris->destino;
sem_post(&S[aris->id]);
```

Figura 3: Semáforo

Terminando se hace un llamado a **VerificarPeso** que se le pasa el camino, pesototal y el mejorpeso del thread, esta función cumple con retornar el valor del mejor peso en el caso de que el thread haya recorrido un mejor peso que algun o algunos de los recorridos anteriores, a su vez verifica si el mejorglobal es peor que el nuevo, en caso de ser peor(mayor) lo cambia.
Cabe recalcar que en el caso de cambiarlo es una Sección Crítica entonces aca se aplica el mutex como se muestra en la figura 4.

```
pthread_mutex_lock(&lock);
if (mejorpeso < mejorglobal)
{
    mejorglobal = mejorpeso;
    caminoglobal = camino;
    cout << "Nuevo mejor peso global: " << mejorglobal << endl;
}
pthread_mutex_unlock(&lock);
```

Figura 4: Mutex

Finalmente se muestra en pantalla el mejor camino encontrado entre todos los threads y cada uno de sus recorridos y el peso. También se muestran los cambios de mejor peso global, en caso de querer ver los mejores pesos de cada thread hay que sacarle un comentario que hay en la función de thread.

```
Nodo : Adyacencia
   A| B-7 C-5 D-1
   B| H-3
   C| H-11
   D| H-10
   E| K-9
   F| K-6
   G| K-9
   H| E-2 F-6 G-7 I-1
   I| K-6
   K|
Nuevo mejor peso global: 26
Nuevo mejor peso global: 23
Nuevo mejor peso global: 22
Nuevo mejor peso global: 17
El mejor camino encontrado es A B  H  I  K  con un peso de 17
```

Figura 5: Output