Avances del Proyecto 2

Clases Agregadas

Para optimizar el trabajo durante el desarrollo, se han identificado diversos comandos de Linux que resultan esenciales para el análisis, la depuración y la optimización de programas. A continuación, se presentan algunas de las herramientas más relevantes, junto con sus respectivas funcionalidades y ejemplos de uso:

1.Interfaz

Esta clase se encarga de centralizar y almacenar toda la información necesaria para mapear los resultados finales. Su diseño está orientado a facilitar la abstracción y la presentación de datos, permitiendo una interacción clara entre los componentes del sistema y la capa de visualización o reporte.

2.Almacén

Atributos

- id: Identificador único asignado a cada almacén.
- capacidad: Indica la cantidad de equipajes actualmente almacenados.
- pais: Cadena de caracteres que representa el país asociado al almacén. Este campo se utiliza para validar la correspondencia con el país de origen del equipaje.
- equipajes: Estructura de tipo cola con prioridad, utilizada para organizar y gestionar el orden de ingreso de los equipajes según criterios preestablecidos.

Funcionalidades Principales

- Constructor de Almacén
  La función de construcción del almacén recorre un arreglo de almacenes, asignando a cada uno un identificador único y estableciendo la capacidad inicial en cero. Esto garantiza que, al inicio de la operación, cada almacén se encuentre en un estado consistente y listo para recibir equipajes.
- almacenar
  Esta función permite agregar un equipaje a la cola de prioridad asociada a un almacén. Al encolar un equipaje, se incrementa el contador de capacidad del almacén, reflejando de forma inmediata la incorporación del nuevo elemento.
- compararPais
  La función *compararPais* determina si un equipaje debe ser asignado a un almacén específico basándose en el país de origen. Su comportamiento se define de la siguiente manera:
  - Caso 1: Si el almacén aún no tiene asignado un país (es decir, el campo pais se encuentra vacío), la función retorna un valor positivo, asignando además el país del equipaje al almacén.

- Caso 2: Si el almacén ya tiene un país definido y éste coincide con el país del equipaje, y la capacidad del almacén se encuentra por debajo del máximo permitido, la función retorna un valor positivo, indicando que el equipaje puede ser almacenado.
- Caso 3: En caso contrario, la función retorna un valor negativo, lo que implica que el equipaje no debe ser encolado en ese almacén.

Proceso Cinta

1. Sincronización Inicial

- Adquisición de semáforos:
  La función inicia esperando la disponibilidad del semáforo semCinta, lo que asegura que el hilo tenga permiso para acceder a la cola de equipajes asignada. Posteriormente, se adquiere el mutex mutexMostrador para obtener acceso exclusivo a la variable compartida cintas.
- Extracción de la cola:
  Se extrae la cola de equipajes correspondiente al índice del hilo (almacenado en cintas[idx]) y se reemplaza por una cola vacía, liberando de forma segura el recurso compartido al liberar el mutex mutexMostrador.

```
void *cinta(void *args){
    int idx = *((int *)args);
    Cola equipaje,vacia;
    int indice = 0,almacenado = 0;
    crear(&vacia);
    while (1){
        sem_wait(&semCinta);
        sem_wait(&mutexMostrador);

        equipaje = cintas[idx];
        cintas[idx] = vacia;

        sem_post(&mutexMostrador);
```

2. Procesamiento de Equipajes

- Recorrido y validación:
  Mientras la cola de equipajes no se encuentre vacía, la función itera sobre los elementos. Por cada equipaje, se recorre la lista de almacenes (hasta MAX_ALMACEN) para determinar si puede ser almacenado. La verificación se realiza mediante la función compararPais, la cual evalúa si el país de origen del equipaje es compatible con el país asignado al almacén y si este cuenta con la capacidad suficiente.
- Almacenamiento seguro:
  Una vez que se identifica un almacén adecuado:
  - Se adquiere el mutex mutexAlmacen para garantizar el acceso exclusivo a la estructura del almacén.

- Se traduce la prioridad del equipaje utilizando la función traducirPrioridad en función del tipo de equipaje.
- Se almacena el equipaje en la cola de prioridad del almacén mediante la función almacenar.
- Finalmente, se libera el mutex mutexAlmacen para permitir que otros hilos puedan acceder a la estructura.

- Actualización de la cola:
  Tras evaluar y, en su caso, almacenar el equipaje, se desencola el elemento procesado y se reinician los contadores y flags necesarios para continuar con el procesamiento del siguiente equipaje.

```
while (esVacio(equipaje) != 1){
    while ((indice < MAX_ALMACEN) && (!almacenado)){
        if(compararPais(equipaje.primero->info.pais,&almacenes[indice])){
            almacenado = 1;
            sem_wait(&mutexAlmacen);

            equipaje.primero->info.prioridad = traducirPrioridad(equipaje.primero->info.tipo);
            almacenar(&almacenes[indice],primero(equipaje)); //encola con prioridad

            sem_post(&mutexAlmacen);
        }
        indice++;
    }
    indice = 0;
    almacenado = 0;
    desencolar(&equipaje);
}
```

3. Condición de Finalización y Liberación de Recursos

- Revisión y salida:
  Una vez procesados todos los elementos de la cola:
  - Se libera el semáforo semCinta, permitiendo que otros hilos puedan acceder a sus respectivas colas.
  - Se adquiere nuevamente el mutex mutexMostrador para verificar el estado de la cola de equipajes.
  - Si se cumple que la cola está vacía y la bandera banderaFinMostrador indica que no se recibirán más maletas, se emite la señal correspondiente y el hilo finaliza su ejecución mediante pthread_exit(NULL).

```
    sem_post(&semCinta);
    sem_wait(&mutexMostrador);

    equipaje = cintas[idx];
    if((!banderaFinMostrador) && (esVacio(equipaje) == 1)){
        sem_post(&mutexMostrador);
        pthread_exit(NULL);
    }

    sem_post(&mutexMostrador);
}
```