Sistemas Operativos
Práctica 1

1. Procesos y comunicación entre procesos

Se pretende realizar un programa que gestione la información de tiempos de viaje entre zonas de Bogotá, almacenada en un archivo csv suministrado por Uber-Movement. Este archivo contiene los siguientes campos:

- sourceid. ID del origen.
- dstid. ID del destino.
- hod. Hora del día.
- mean_travel_time. Media de tiempo de viaje.
- standard_deviation_travel_time
- geometric_mean_travel_time
- geometric_standard_deviation_travel_time

Al ejecutar el programa se muestra el siguiente menú, solicitando la opción y *enter*:

```
Bienvenido

1. Ingresar origen
2. Ingresar destino
3. Ingresar hora
4. Buscar tiempo de viaje medio
5. Salir
```

En cada opción se presentará la siguiente información:

1. **Ingresar origen**. Al ingresar Solicita el ID del origen y se espera un número entero entre 1 y 1160. El usuario deberá digitar enter para ingresar el número. Por ejemplo:

   ```
   Ingrese ID del origen: 105
   ```

2. **Ingresar destino**. Al ingresar Solicita el ID del destino y se espera un número entero entre 1 y 1160. El usuario deberá digitar enter para ingresar el número. Por ejemplo:

   ```
   Ingrese ID del destino: 1015
   ```

3. **Ingresar hora del día**. Al ingresar Solicita la hora del día. Se espera un número entero entre 0 y 23. El usuario deberá digitar enter para ingresar el número. Por ejemplo:

   ```
   Ingrese hora del día: 22
   ```

4. **Buscar tiempo de viaje medio**. Al ingresar se inicia la búsqueda y una vez se ha encontrado se presenta en la pantalla. Por ejemplo:

   ```
   Tiempo de viaje medio: 1268.8
   ```

   En caso de no encontrar el valor, se muestra el mensaje "NA".

5. **Salir**.

1.1. Consideraciones.

- Hacer uso de punteros y de memoria dinámica (malloc() - free()).
- Si genera un nuevo archivo indexado, se recomienda almacenar estructuras de forma binaria, no texto. El archivo deberá residir en disco, los datos **NO** deben estar en memoria. La cantidad de memoria empleada por el proceso no deberá superar 1MB.
- La búsqueda deberá ejecutarse en un tiempo menor a 2 segundos.
- Se deberán implementar dos procesos, uno para la apertura y búsqueda dentro del archivo y otro para la interfaz de usuario. La comunicación se deberá realizar mediante tuberías nombradas o memoria compartida. Los procesos deben ser **no** emparentados.
- Se debe implementar una tabla *hash* para la búsqueda dentro archivo. Se recomienda indexar el origen y generar una lista enlazada por cada ID de origen dentro de un nuevo archivo. Esto asegura que se busca un origen-destino solo en la lista correspondiente al origen (ver figura).
- Entrega: Archivo fuente con *main* en **p1-odProgram.c** (se pueden tener más archivos), archivo **Makefile** para compilar todo, archivo LEEME dentro de una carpeta con los nombres que aparecen en el correo para cada integrante. Pej: **capedrazab-capedrazab**. Este archivo o carpeta se entregará en clase.

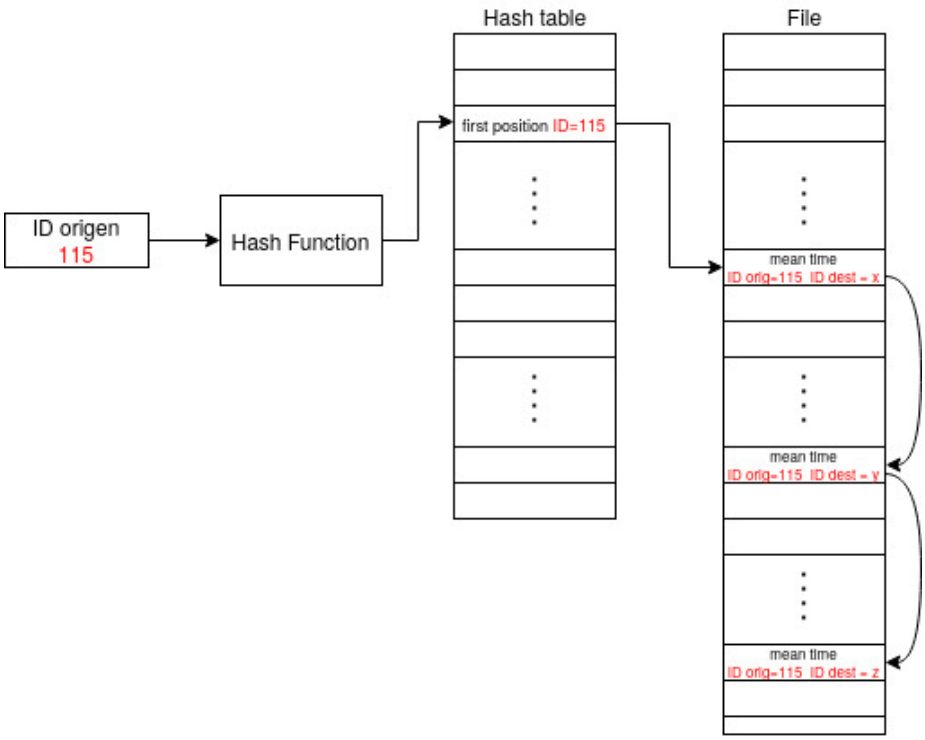

Figura 1: .

1.2. Calificación.

Se tendrán en cuenta los siguientes aspectos para la evaluación:

- Funcionamiento del programa. 40 %
- Rapidez de la búsqueda (tabla hash, búsqueda <2 segundos). 10 %
- Código limpio. 10 % (modular, tabulaciones, comentarios - básicos, declaración de constantes, etc.)
- Sustentación (se selecciona a cualquier integrante): 20 %.
- Código en repositorio git (github u otro) 10 %
- Uso de herramientas como copilot, chatGPT, otras. Se demuestra en la sustentación. 10 %