

Autores:

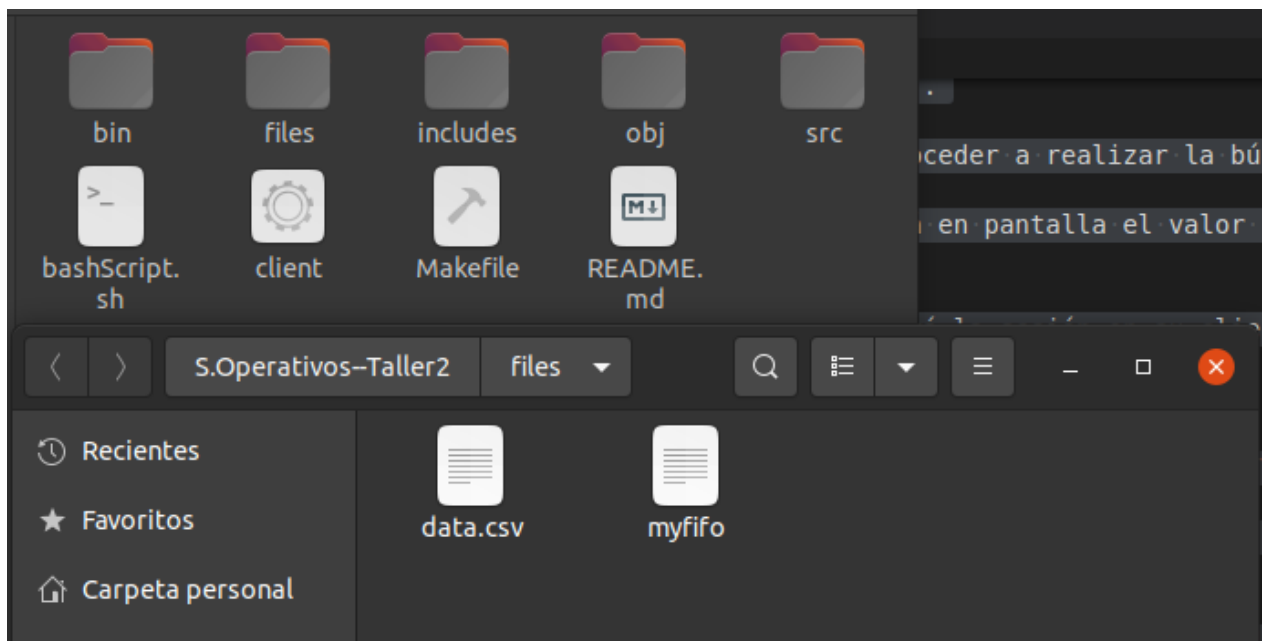
- Santiago Rodríguez Vallejo - 100106037 - sarodriguezva@unal.edu.co - Ingeniería de Sistemas y Computación
- Cristhian David Mora Uribe - 1233494036 - cdmorau@unal.edu.co - Ingeniería de Sistemas y Computación
- Javier Esteban Pacavita Galindo - 1015478836 - jpacavita@unal.edu.co - Ingeniería de Sistemas y Computación

Asignatura: Sistemas Operativos.

Tema: Taller 2. Documentación.

MANUAL DE USO:

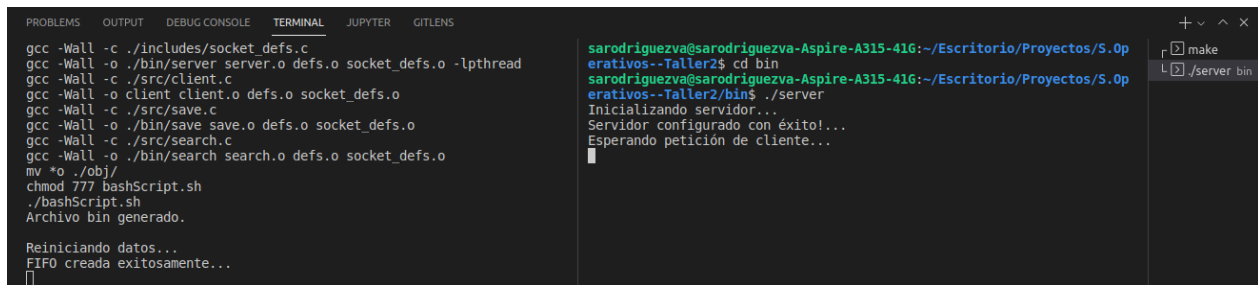
1. Crear una carpeta "files" y poner dentro el archivo de Uber con el nombre "data.csv".



2. Abrir un terminal en la carpeta del proyecto. (Terminal 1)
3. Abrir un terminal en la carpeta ./bin. (Terminal 2)
4. En el Terminal 1, ejecutar el comando make. El comando generará internamente el archivo en disco de medias de viaje. Una vez hecho esto ejecutará el servicio de búsqueda a la escucha de datos de entrada provenientes del servidor.

5. Esperar hasta ver el mensaje "Obteniendo Data..."
6. En el Terminal 2, ejecutar el comando `./server` El comando configurará internamente el servidor y lo pondrá a la escucha de conexiones con clientes.
7. Esperar hasta ver el mensaje "Esperando petición del cliente..."

VISUALIZACIÓN TERMINALES 1 y 2



The image shows two terminal windows from a code editor. The left terminal shows the compilation of a C program using `gcc`. It compiles `socket_defs.c`, `server.c`, `client.c`, `save.c`, and `search.c` into object files and then links them into an executable `bin`. The right terminal shows the execution of `./server`, which outputs "Iniciando servidor...", "Servidor configurado con éxito!", and "Esperando petición de cliente...".

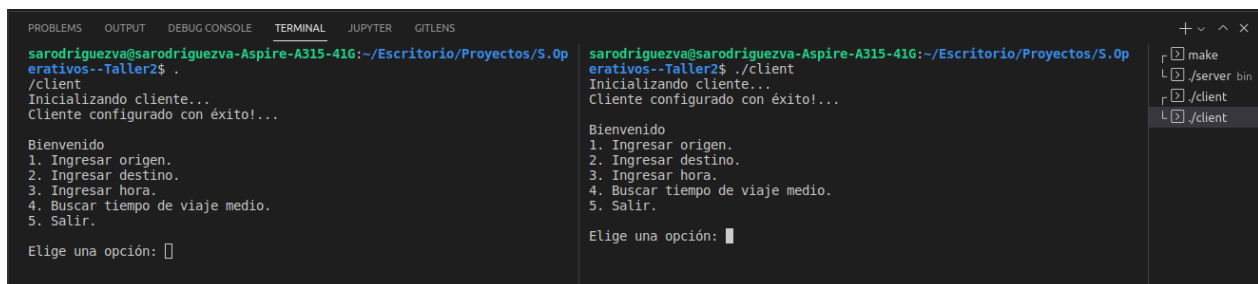
```
gcc -Wall -c ./includes/socket_defs.c
gcc -Wall -o ./bin/server server.o defs.o socket_defs.o -lpthread
gcc -Wall -c ./src/client.c
gcc -Wall -o client client.o defs.o socket_defs.o
gcc -Wall -c ./src/save.c
gcc -Wall -o ./bin/save save.o defs.o socket_defs.o
gcc -Wall -c ./src/search.c
gcc -Wall -o ./bin/search search.o defs.o socket_defs.o
mv *o ./obj/
chmod 777 bashScript.sh
./bashScript.sh
Archivo bin generado.

Reiniciando datos...
FIFO creada exitosamente...
```

```
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$ cd bin
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2/bin$ ./server
Iniciando servidor...
Servidor configurado con éxito!...
Esperando petición de cliente...
```

8. Abrir un terminal en la carpeta del proyecto por cada cliente nuevo que se desea conectar. (Terminal n).
9. En el Terminal n, ejecutar el comando `./client`
10. Esperar a ver el menú de opciones.

VISUALIZACIÓN DE DOS CLIENTES



The image shows two terminal windows. The left terminal shows the execution of `./client`, which outputs "Iniciando cliente...", "Cliente configurado con éxito!", and a menu of options: "Bienvenido", "1. Ingresar origen.", "2. Ingresar destino.", "3. Ingresar hora.", "4. Buscar tiempo de viaje medio.", "5. Salir.", followed by "Elige una opción:". The right terminal shows the execution of `./client` again, displaying the same menu of options.

```
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$ ./client
Iniciando cliente...
Cliente configurado con éxito!...

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

Elige una opción: 
```

```
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$ ./client
Iniciando cliente...
Cliente configurado con éxito!...

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

Elige una opción: 
```

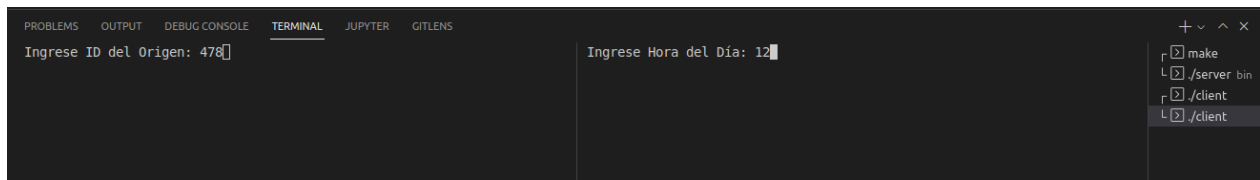
11. Puede utilizar la interfaz libremente, con la siguiente guía: La interfaz muestra 5 posibles opciones,

1. Ingresar origen.
2. Ingresar destino.
3. Ingresar hora.

4. Buscar tiempo de viaje medio
5. Salir.

El sistema tratará cada opción que elija a la vez. Entre las opciones 1 y 3 pedirá que ingrese los valores que desea como origen, destino y hora de viaje correspondientes para realizar la búsqueda en el sistema. De no cargar algún dato, este internamente será tratado como cero (0).

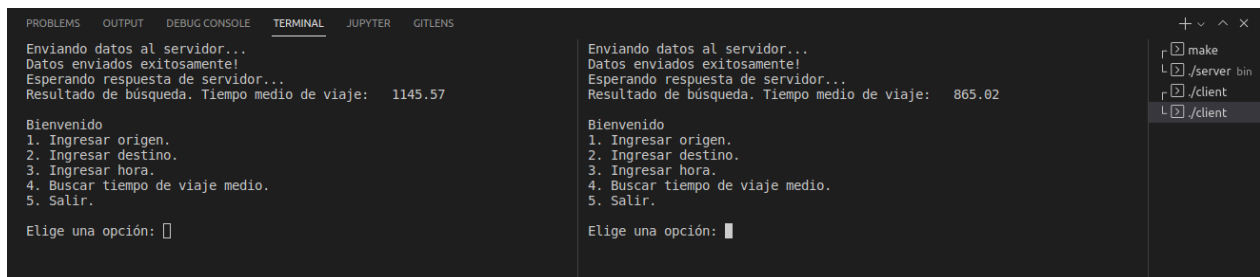
EJEMPLO



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER  GITLENS
Ingrese ID del Origen: 478
Ingrese Hora del Día: 12
```

Una vez cargados los 3 datos, puede proceder a realizar la búsqueda de su valor deseado al seleccionar la opción 4. Al seleccionar esta opción se imprimirá en pantalla el valor correspondiente al viaje medio según sus datos.

EJEMPLO



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER  GITLENS
Enviando datos al servidor...
Datos enviados exitosamente!
Esperando respuesta de servidor...
Resultado de búsqueda. Tiempo medio de viaje: 1145.57

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

Elige una opción:

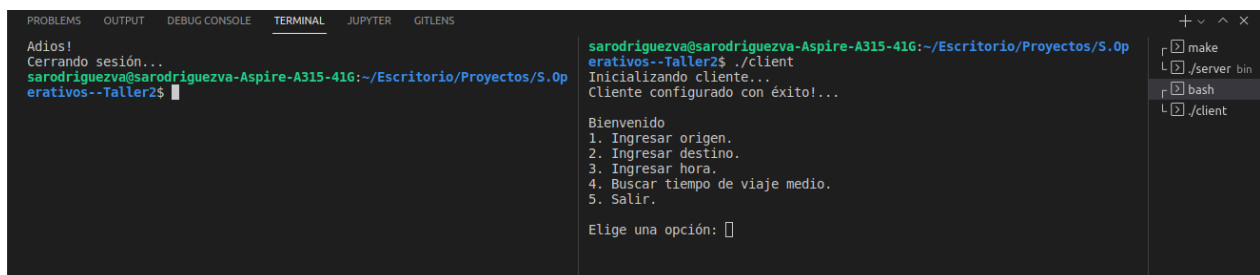
Enviando datos al servidor...
Datos enviados exitosamente!
Esperando respuesta de servidor...
Resultado de búsqueda. Tiempo medio de viaje: 865.02

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

Elige una opción:
```

Al seleccionar la opción 5, se terminará la sesión en su cliente.

EJEMPLO



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER  GITLENS
Adios!
Cerrando sesión...
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$
sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$

sarodriguezva@sarodriguezva-Aspire-A315-416:~/Escritorio/Proyectos/S.Operativos--Taller2$ ./client
Iniciando cliente...
Cliente configurado con éxito!...

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

Elige una opción:
```

Notas:

* HASTA 32 CLIENTES PUEDEN CONECTARSE A LA VEZ EN EL SISTEMA.

* El servidor imprimirá los registros de búsquedas hechos por cada cliente durante su tiempo de servicio.

EJEMPLO

```
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2$ cd bin
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2/bin$ ./server
Iniciando servidor...
Servidor configurado con éxito!...
Esperando petición de cliente...
Conexión con cliente recibida!...
Esperando petición de cliente...
Conexión con cliente recibida!...
Esperando petición de cliente...
Cliente desconectado!
[Fecha 2022-06-22 T10:16:25] Cliente [127.0.0.1] [865.02 - 951 - 151]
```

12. Una vez terminada la ejecución del proyecto, se recomienda terminar a mano los programas de los Terminales 1 y 2 para evitar el consumo de recursos en segundo plano.

EJEMPLO

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS
Adios!
Cerrando sesión...
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2$

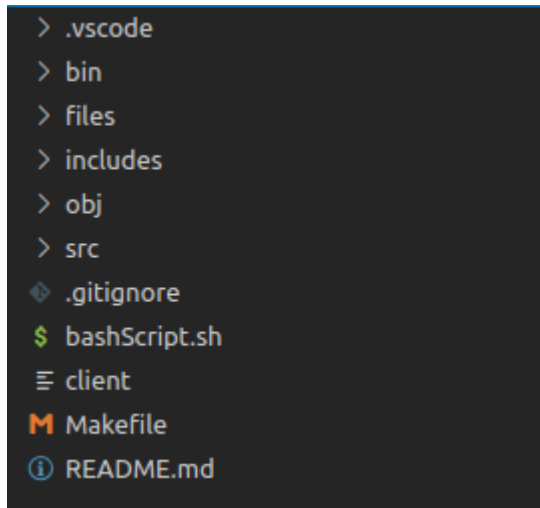
erativos--Taller2$ cd bin
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2/bin$ ./server
Iniciando servidor...
Servidor configurado con éxito!...
Esperando petición de cliente...
Conexión con cliente recibida!...
Esperando petición de cliente...
Conexión con cliente recibida!...
Esperando petición de cliente...
Cliente desconectado!
[Fecha 2022-06-22 T10:16:25] Cliente [127.0.0.1] [865.02 - 951 - 151]
Cliente desconectado!
^C
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2$
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS
Datos obtenidos:
Origen: 951
Destino: 151
Hora: 12

Buscando valor...
Indice encontrado! Indice: 16322150
DATA BIN abierto con éxito...
Valor encontrado! Valor: 865.02
Enviando valor...
Valor enviado con éxito!
Reiniciando datos...
^Cmake: *** [Makefile:62: run] Interrupción
sarodriguezva@sarodriguezva-Aspire-A315-41G:~/Escritorio/Proyectos/S.Op
erativos--Taller2$
```

INFORME DE ELABORACIÓN

Para el proyecto se establece la siguiente estructura de directorios y archivos:



Dónde en src se encuentran los archivos *.c en dónde se encuentra el core del sistema.

Se empieza por la definición de un par de cabeceras para compartir entre archivos. Dichas cabeceras se ubican en el directorio ./includes/ y son las siguientes:

- defs.h
 - Incluye constantes globales de rutas a archivos relevantes en el proyecto y la estructura de viajes.
- socket_defs.h
 - Incluye la definición del puerto y el backlog como constantes globales y la estructura sock_addr_in.

Ambas cabeceras contienen la inclusión de librerías necesarias para el funcionamiento de los programas.

Se detallan aquellos prototipos de funciones definidas en dichas cabeceras, cuya implementación se realiza respectivamente en defs.c y socket_defs.c:

- defs:
 - int hash(int origen, int destino, int hora);
 - void getTimeStamp();
- socket_defs
 - int createSocket(int fd);
 - int configureServerSocket(int fd, struct sock_addr_in *addr);
 - void configureClientSocket(struct sock_addr_in *addr, char *ip_addr);
 - int setSocketToListen(int fd);
 - int connectSocket(int fd, struct sock_addr_in *addr);

La implementación de todas las funciones en el proyecto incluye sus debidos comentarios y documentación explicatoria.

En la carpeta `./src/` se tienen 4 archivos `*.c` definidos, los cuales se enlistan a continuación incluyendo el conjunto de funciones definidas para cada uno:

- `save.c`
 - `void loadArray(double* result_list, viaje tmp);`
 - `void loadBinFile(double* result_list);`
 - `int main();`
- `search.c`
 - `void resetValues(int *origen, int *destino, int *hora);`
 - `void getValues(int fd, int *origen, int *destino, int *hora);`
 - `double search(int origen, int destino, int hora);`
 - `void sendData(int fd, double data);`
 - `int main();`
- `server.c`
 - `void *processClient(void *pclient_fd);`
 - `void receiveDataFromClient(int client_fd, int *origen, int *destino, int *hora);`
 - `double search(int *origen, int *destino, int *hora);`
 - `void printLog(struct sock_addr_in address, int origen, int destino, int hora, double tiempo_viaje);`
 - `void stop();`
 - `int main();`
- `client.c`
 - `int showInterface(int *fd, int *origen, int *destino, int *hora, double *tiempo_viaje);`
 - `void getValue(int *var, char *message, int min, int max);`
 - `void sendDataToServer(int fd, int *origen, int *destino, int *hora);`
 - `int main();`

En corto, **save.c** se encarga de hashear los registros del archivo de Uber y guardarlos en disco duro para facilitar la búsqueda. Es el primer proceso en ejecutarse y es un requisito para el funcionamiento correcto del sistema.

search.c es un proceso “servicio” de búsqueda, el cual recibe los datos de **server.c** por medio de una tubería nombrada y se encarga de realizar la respectiva búsqueda en el archivo generado por `save.c` para devolver el valor buscado por la misma tubería a **server.c**.

server.c es el proceso controlador principal, se encarga de aceptar y crear un hilo de ejecución por cada cliente conectado, para cada uno recibe los datos por medio de una conexión por sockets, y los datos recibidos los envía a **search.c** por medio de una tubería nombrada, también recibe el valor que el cliente busca desde **search.c** y se lo devuelve al mismo cliente por el socket.

client.c es el proceso de comunicación frente a frente con el cliente, se encarga de crear el socket y conectarlo con el servidor para después mostrar la interfaz y recibir los datos por teclado, gestionando las acciones para cada opción que el cliente elija.

Todos los archivos en el proyecto se compilan adecuadamente por medio de un **MakeFile**, y este también llama a la ejecución de un archivo **bashScript.sh** que se encarga de llamar los procesos **save** y **search**.

DIAGRAMAS

