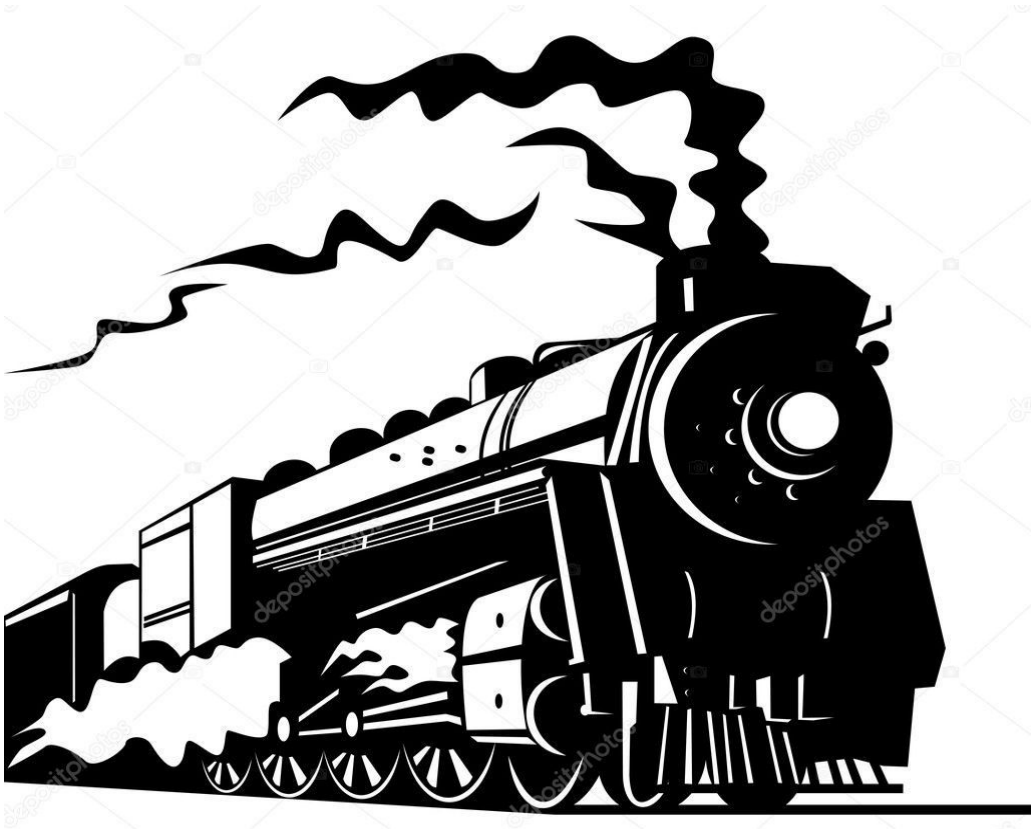
Sistemas Operativos

Trabajo Práctico 2019

Trenes



Profesor: Ing. Mostovoi Alejandro

Integrantes del Grupo:
Fernandez Gonzalo
Palacios Facundo
Pignatta Mauro
Pirrota Ezequiel
Ugobono Alejandro
Wiñar Mariano

Índice

Funciones de Estación

Encontraremos aquí todas aquellas funciones necesarias para la gestión, proceso y administración de todas las estaciones, en sus facetas tanto como tipo servidor hacia los clientes, es decir la comunicaciones entre servidor/cliente (proceso estación/proceso tren), como así también todas aquellas que tienen que ver con al operatoria y administración de cada estación

1- Estructuras de Datos:

1.1- Estructura Estaciones

```
typedef struct
{
        int ID;
        char nombre[max_nombre_est];
        int distancia;
        int online;
        int nCliente;
        TREN tren[MAX_TREN];
}ESTACION;
```

1.2- Estructura Nodo Trenes

```
typedef struct nodo
{
        struct nodo * sig;
        TREN * tren;
        int prioridad;
}ST_NODO_TRENES;
```

1.3- Variables global de Punteros a Tren, Nodo Cola Prioridad Menor y Mayor

```
pthread_mutex_t lock;
TREN * anden ;
ST_NODO_TRENES * ColaPrioridadMenor ;
ST_NODO_TRENES * ColaPrioridadMayor ;
```

1.4- Variable Global Puntero tipo File

```
pthread_mutex_t log_lock;
FILE * logEstacion;
```

1.5- Variable global del servidor de estaciones con su máximo de estaciones

```
int serverEst[MAX_ESTACION];
```

1.6- Variable global estaciones con el tipo de estructura de estaciones y su cantidad máxima definida:

```
ESTACION estaciones[MAX_ESTACION];
```

1.7- Variable global de un entero que se utiliza para las posiciones
int miPos;
1.8- Variable global de un entero para el tren viajando
int trenEnViaje;

2- Funciones y Aplicaciones:

2.1 int ObtenerDatosMiEstacion(char * nomArchivo, ESTACION est[]):
Aplicación:
Abre el archivo de configuración pasado como argumento y lo guarda, en la posición del vector de estaciones que corresponda,
Parámetros:
* nomArchivo es el nombre del archivo.
ESTACION est es el vector de estaciones,
Retorna:
Devuelve la posición en la que se encuentra tu estación.

2.2 void ObtenerOtrasEstaciones(ESTACION est[],int miPos):
Aplicación:
Abre el resto de los archivos de configuración, y obtiene los datos de las demás estaciones.
Parámetros:
ESTACION est es el vector de estaciones,
int miPos guarda la posición de la estación en el vector.

2.3 int registrarTren(ESTACION *estacion, char * mensaje):
Aplicación:
Copia los datos del tren en la estación en caso que haya lugar disponible,
Parámetros:
ESTACION * estacion puntero a la estación,
char * mensaje contiene los datos para registrar el tren,
Retorna:
Devuelve 1 si el tren se registro correctamente y 0 en caso de que no se haya registrado.

2.4 int buscarTrenes(TREN trenes[] ,int posTrenes[]):
Aplicación:
Devuelve un vector con las posiciones del vector de trenes en las que se encuentran,
Parámetros:
TREN trenes es el vector de trenes,
int posTrenes indica la posición de los trenes, encontrados,
Retorna:
Devuelve la cantidad.

2.5 int BuscarTrenPorID(ESTACION estacion, int idTren):
Aplicación:
Busca un Tren en el vector de trenes,
Parámetros:
ESTACION estacion variable del tipo ESTACION,
int idTren es el numero de tren a buscar,
Retorna:
Devuelve la posición en la que se encuentra el tren en el vector de trenes de la estación, o -1 si no se encuentra.

2.6 void ConexionServer():
Aplicación:
Para el hilo que se encarga de la conexión servidor-cliente.

2.7 int buscarEstacionPorNombre(char * mensaje):
Aplicación:
Busca a la estación por el nombre,
Parámetros:
char * mensaje donde tiene el nombre de la estación a buscar,
Retorna:
Devuelve la pos si la encuentra o -1 si no la encuentra.

2.8 int mensajeListadoEstDisp(char * mensaje):
Aplicación:
Copia las estaciones disponibles para viajar,
Parámetros:
Mensaje puntero a char donde se van a copiar las estaciones disponibles,
Retorna:
Devuelve la cantidad de estaciones que están disponibles para viajar.

2.9 int mostrarTrenesMigrados(char * mensaje):
Aplicación:
Copia los trenes que migraron al mensaje*,
Parámetros:
Mensaje puntero a char donde se van a copiar los trenes,
Retorna:
Devuelve la cantidad de trenes migrados.

2.10 int elegirTren():
Aplicación:
Pide al usuario que ingrese el tren que quiere que viaje,
Retorna:
Posición del tren elegido en el vector o -1 en caso de que el tren elegido no sea valido.

2.11 int elegirEstDestino():
Aplicación:
Pide al usuario que ingrese la estación donde quiere viajar*,
Retorna:
Posición la estación elegida o -1 en caso de que no sea valida.

2.12 int calcularTiempoDeViaje(int posEstacionDestino):
Aplicación:
Calcula el tiempo basándose en la distancia entre una estación y otra,
Parámetros:
posEstacionDestino La posición en el vector de estación a la cual se quiere dirigir el tren,
Retorna:
Tiempo de viaje.

2.13 void prepararEnvioTren(char *mensaje , TREN * tren):
Aplicación:
Prepara el mensaje para enviar un tren de una estación a otra,
Parámetros:
* mensaje puntero a char copia el mensaje a enviar,
* tren Para saber que tren hay que enviar.

2.14 ST_NODO_TRENES * crearNuevoNodo(TREN * tren):
Aplicación:
Crea nodo de tren,
Parámetros:
* puntero tren de estructura TREN,
Retorna:
Devuelve el nodo del nuevo nodo creado.

2.15 void encolarTren(TREN * tren, ST_NODO_TRENES ** cola):
Aplicación:
Suma trenes a la cola de trenes,
Parámetros:
* puntero tren de estructura TREN,
** cola puntero a puntero del Nodo Trenes.

2.16 TREN * asignarAnden(ST_NODO_TRENES ** cola):
Aplicación:
Asigna Anden al tren,
Parámetros:
** cola puntero a puntero del Nodo Trenes,
Retorna:
Devuelve el puntero del tren afectado.

2.17 TREN * eliminarNodoPrioridad(ST_NODO_TRENES ** cola):
Aplicación:
Elimina Nodos según la prioridades,
Parámetros:
** cola puntero a puntero del Nodo Trenes,
Retorna:
Devuelve el puntero del tren afectado.

2.18 TREN * eliminarNodoTrenSegunID(int IDTren, ST_NODO_TRENES ** cola):
Aplicación:
Elimina Nodo del tren por ID,
Parámetros:
int IDTren para identificar al tren afectado,
** cola puntero a puntero del Nodo Trenes,
Retorna:
Devuelve el puntero del tren afectado.

2.19 int subirPrioridadTrenes(ST_NODO_TRENES * cola):
Aplicación:
Sube la prioridad a los trenes por tiempo de su espera en la cola,
Parámetros:
* cola puntero al Nodo Trenes,
Retorna:
Devuelve el entero con la sumatoria que le corresponda según la vuelta de prioridades.

2.20 void CambiarDeColaTrenes(ST_NODO_TRENES ** cola_Menor, ST_NODO_TRENES ** cola_Mayor, int cantNodos):
Aplicación:
Cambia de cola menor prioridad a la Cola Mayor prioridad,
Parámetros:
** cola Menor puntero a la Colas de Menor prioridad,
** cola Mayor puntero a la Colas de Mayor prioridad,
int cantNodos lleva la cantidad de nodos tipo entero.

2.21 void NuevoTrenAnden(TREN ** anden, ST_NODO_TRENES ** cola_Menor, ST_NODO_TRENES ** cola_Mayor):
Aplicación:
Nuevo tren para anden,
Parámetros:
TREN ** anden puntero de puntero anden de trenes,
** cola Menor puntero a la Colas de Menor prioridad,
** cola Mayor puntero a la Colas de Mayor prioridad.

2.22 FILE * crearLogEstacion(char * nombreEstacion):

Aplicación:

Crea el Archivo tipo txt según nombre de la estación,

Parámetros:

char * nombreEstacion puntero del nombre de la estación,

Retorna:

Puntero tipo FILE al archivo txt de la estación.

2.23 void * llenarLog(TREN * tren , FILE * logEstacion):

Aplicación:

Carga los datos en el archivo txt de la estación afectada,

Parámetros:

TREN * tren puntero estructura tipo TREN con los datos del tren afectado,
FILE * logEstacion puntero tipo FILE al archivo txt de la estación afectada.

2.24 void avisarEstaciones(int posEstacionDestino, int tipoAviso);

Aplicación:

Avisa a las estaciones que hay un tren en viaje o que llego al destino.

Parámetros:

int posEstacionDestino posición de la estación a la que viaja el tren
int tipoAviso es 1 para avisar que viaja, 2 para avisar que llego.