Índice

1. Introducción

Se implementó un sistema cliente-servidor para jugar al Sudoku de manera remota. Para ello se desarrolló un programa que puede ejecutarse en modo cliente o servidor.
La motivaciones principales y la búsqueda del **Trabajo Práctico 1** son:

- El aprendizaje del uso de *sockets* para establecer conexiones entre máquinas remotas.
- La aplicación de buenas prácticas en programación de Tipos de Datos Abstractos.
- La modularización criteriosa de los sistemas.
- Correcto uso y liberación de recursos.

2. Cuestiones de implementación

2.1. Descripción de los TDAs implementados

A continuación, mencionaremos y explicaremos brevemente los **TDAs** involucrados en la implementación del presente trabajo.

- **socket_t** Implementa las funciones principales de un *socket*: enviar y recibir mensajes, conectarse a un host en un determinado puerto, enlazarse a un puerto, aceptar clientes, liberar los recursos utilizados por el *socket*, etcétera. Cabe aclarar que el TDA *socket* implementado responde a los protocolos **TCP** e **IPv4**.
- **client_socket_t** Implementa las acciones propias de un *socket* cliente (enviar y recibir mensajes, conectarse a un servidor, etcétera). Para implementarlo, se utilizaron las funciones implementadas en el TDA **socket_t**.
- **server_socket_t** Implementa las acciones propias de un *socket* servidor (enviar y recibir mensajes, enlazarse sobre un puerto y "escuchar" conexiones entrantes sobre el mismo, aceptar nuevos clientes, etcétera). Cabe aclarar que se implementó tal que el *socket* pueda aceptar una única conexión y atender sus pedidos hasta que el cliente la cierre remotamente. Para implementarlo, se utilizaron las funciones implementadas en el TDA **socket_t**.
- **cell_t** Implementa las acciones propias de una celda de un tablero de Sudoku. Se inicializa con un número del 0 al 9 (en la implementación propuesta, inicializarla con un cero representa que se inicializa vacía), se le puede setear un número del 1 al 9, se la puede resetear, se puede obtener el número que contiene y se puede comparar con otra celda para saber la validez de la unión entre las mismas.
- **sudoku_board_t** Implementa las acciones propias de un tablero de sudoku. Puede insertar un número en una fila y columna determinadas, puede verificar si se están cumpliendo las reglas, puede resetear el tablero (sin modificar el contenido de las celdas *vía*) y puede mostrar sus celdas.
- **client_protocol_t** Implementa el protocolo establecido para la comunicación entre el cliente y el servidor. Se encarga de operar los comandos ingresados por el usuario para enviar el comando a través del socket cliente tal que el servidor lo entienda y responda según el protocolo propuesto.
- **server_protocol_t** Recibe un mensaje proveniente de la máquina cliente, lo interpreta para realizar una acción sobre el tablero y responde con un mensaje al cliente (enviándolo a través del socket servidor).
- **client_interface_t** Engloba a varios de los TDAs para el cliente ya mencionados, coordinando su funcionamiento para procesar comandos ingresados por el usuario secuencialmente.

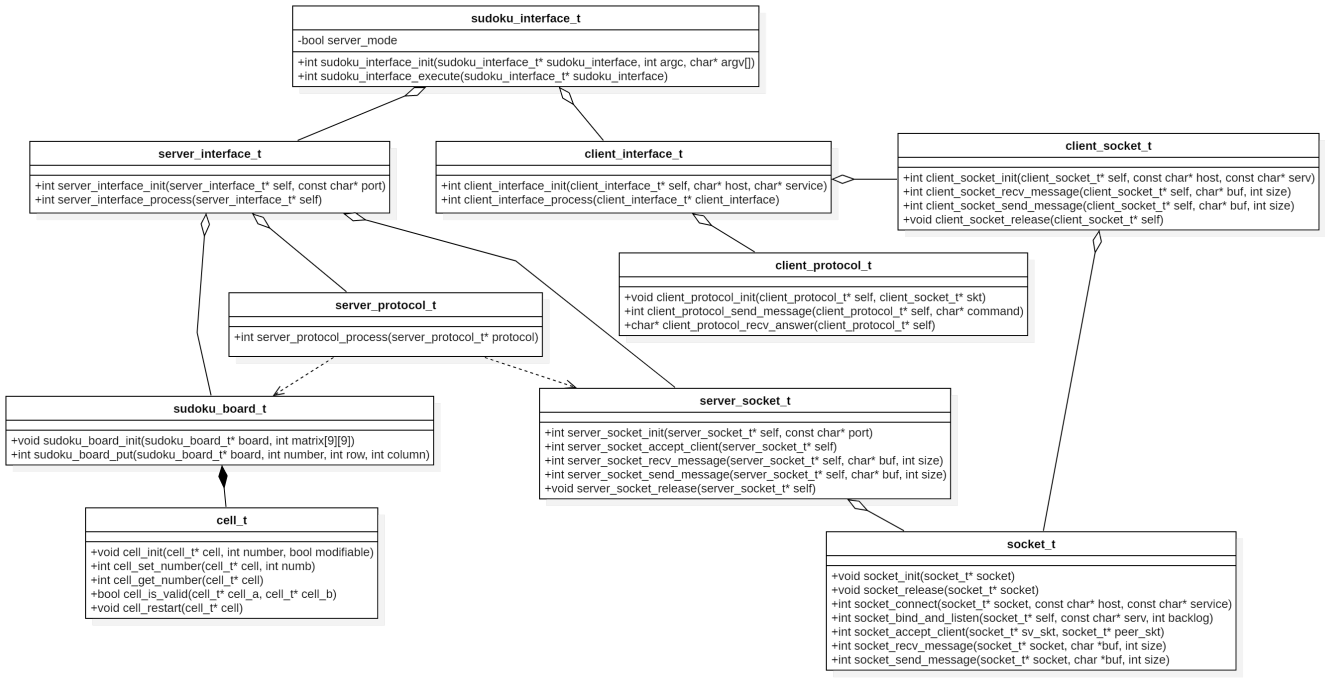
- **server_interface_t** Engloba a varios de los TDAs para el servidor ya mencionados, coordinando su funcionamiento para recibir mensajes provenientes a través del socket cliente y procesarlos (nos referimos a procesar como la acción de interpretar un mensaje, tomar una acción debido al mismo y responder al cliente según lo que corresponda).
- **sudoku_interface_t** Engloba el funcionamiento del **Sudoku Remoto** iniciando el juego en modo **servidor** o **cliente** y haciendo uso de los TDAs server_interface_t y client_interface_t para su ejecución. Además, verifica que los argumentos con los que se invoca el ejecutable sean correctos.

2.2. Mención a los módulos de los que hacen uso algunos TDAs

- **protocol_message_maker** Este módulo es utilizado por el TDA **client_protocol_t** para armar el mensaje a enviarle al servidor según el comando ingresado por entrada estándar.
- **board_representation_maker** Este módulo es utilizado por el TDA **server_protocol_t** para obtener una representación gráfica del estado actual del tablero. Esta representación creada por este módulo, se envía a través del *server socket* a la máquina cliente.
- **sudoku_matrix_loader** Este módulo es utilizado por el TDA **server_interface_t** para cargar la matriz contenida en el archivo *board.txt* en una matriz auxiliar y así poder inicializar el tablero del Sudoku.

3. Representación del diseño implementado

3.1. Diagrama de clases



4. Conclusión

4.1. Desafíos enfrentados

A la hora de hablar del diseño, creemos que el desafío más grande del presente trabajo fue la modularización criteriosa del código para evitar acoplar por demás los distintos subsistemas que lo componen y que, además, estos sean altamente cohesivos.
Lograr esto no resultó nada sencillo, a medida que se avanzó en el código, se encontraron errores de diseño -tales como altas dependencias entre diversos módulos- que se corrigieron definiendo claramente las responsabilidades de cada bloque y no añadirle complejidad más allá de ellas.
Refiriéndonos al código, no fue trivial la implementación del protocolo pedido junto con el correcto envío de los distintos mensajes entre el cliente y el servidor. Esto se debió en gran parte a que el *debugging* se complejizaba exponencialmente al aumentar las dependencias entre los distintos subsistemas del programa.

4.2. Funciones que resultaron útiles

1. **memcpy** De gran ayuda para *castear* variables al tipo de dato **uint32_t**. Esto es sumamente necesario para la utilización de las funciones **htonl** y **ntohl**.

2. **sscanf** Realmente interesante y útil para parsear *strings*. Fue de gran ayuda a la hora de parsear la entrada estándar para la verificación de los argumentos.