
C
Talks

Talk I
Basis & Best Practices

Temario

- ANSI C for Dummies
- ***TAD***
- Best Programming Practices

TAD
Tipo Abstracto de Dato

(From UTN-FRBA SSL Modulo IV K1GT5 - pag. 2)

Un **Tipo de Dato** es un conjunto de ***valores*** más un conjunto de ***operaciones*** aplicables sobre estos valores. Se clasifican en:

- ***Tipos de Datos Primitivos***: provistos por el lenguaje.

  Ej: **int** y sus operaciones son: +, -, *, /, %, <, >, !=, etc.

- ***Tipos de Datos Abstractos (TADs)***: creados por el programador (estructura y operaciones).

  Ej: Pila, Cola, Lista, Persona.

  Y operaciones como: meter y sacar, agregar y quitar,

  insertar y suprimir, saludarA.

- ***Tipos de Datos Semi-Abstractos***: el lenguaje brinda algunas herramientas para construir el Tipo, y las operaciones en general las debe implementar el programador.

  Ej: strings. Y operaciones como: strcmp, strcpy, strcat.

Ahh! ¿¡Eso era un TAD!?

Un TAD permite abstraernos de la implementación de los datos. La idea básicamente es **separar el uso del Tipo de Datos de su Implementación**.

A ésto se lo conoce como **Abstracción de Datos**.

Se centra en la **Modularización** y el **Encapsulamiento**.

Permite la re-utilización de código y el diseño e implementación de Bibliotecas propias.

Hay 3 pasos a seguir para crear un TAD:

1. Especificación
2. Diseño
3. Implementación

1. Especificación

Persona_crear :: Cadena x Cadena x Char x int -> Persona*

Persona_matar :: Persona* ->Nada

Persona_equals :: Persona* x Persona* -> int

Persona_saludarA :: Persona* -> Nada

Persona_cumplirAños :: Persona* -> Nada

Persona_esJubilado :: Persona* -> int

Persona_maldecir :: Nada -> Nada

Persona_falsificarIdentidad :: Persona* x Cadena x Cadena -> Persona*

2. Diseño

En esta etapa se crea la **Interfaz** del TAD: el **header** (.h) (archivo encabezado).

La Interfaz es la **Parte Pública** de un TAD, la que posee los prototipos de las funciones y la declaración del TAD.

Con sólo ver la Interfaz uno puede **comprender** que hace ese TAD, y de ser necesario, se apoya en la documentación (si existe alguna).

3. Implementación

Finalmente, se comienza con la codificación de las funciones, es decir, la implementación del TAD. Ésto se hace en un **Source File**, un **.c**

La Implementación es la **Parte Privada** de un TAD, la que posee los **detalles** de lo que hace cada función.

Para el usuario del TAD ésto es transparente, ya que puede simplemente llamar a las funciones tantas veces como quiera, **abstrayéndose de la implementación** de las mismas.

3. Implementación

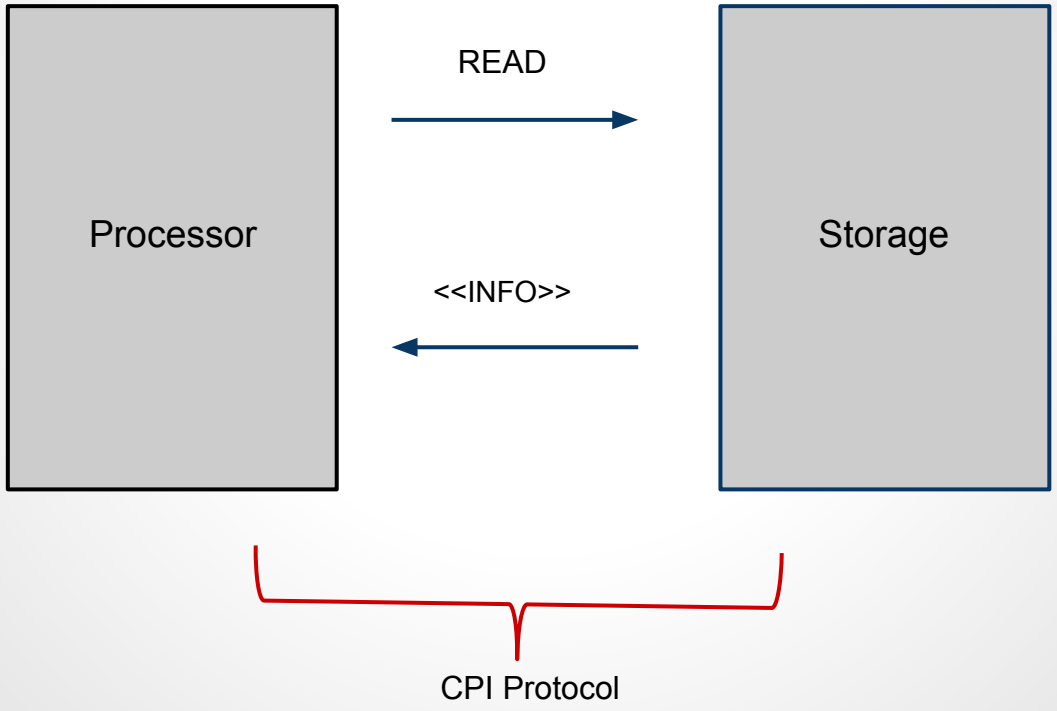
Para **una Interfaz**, pueden existir **muchas implementaciones** distintas.

El hecho de trabajar con un header a nivel diseño, nos permite tener distribuir la Parte Pública de nuestro TAD, para que cada programador lo implemente a su manera.

Es tan simple como cambiar un Source File por otro.

Caso de Prueba I

Un proceso llamado "Processor", se comunica con un proceso "Storage" mediante un protocolo de comunicación red propio llamado CPI. El "Processor" envía paquetes cuya codificación indican si se quiere leer, escribir, borrar o actualizar.

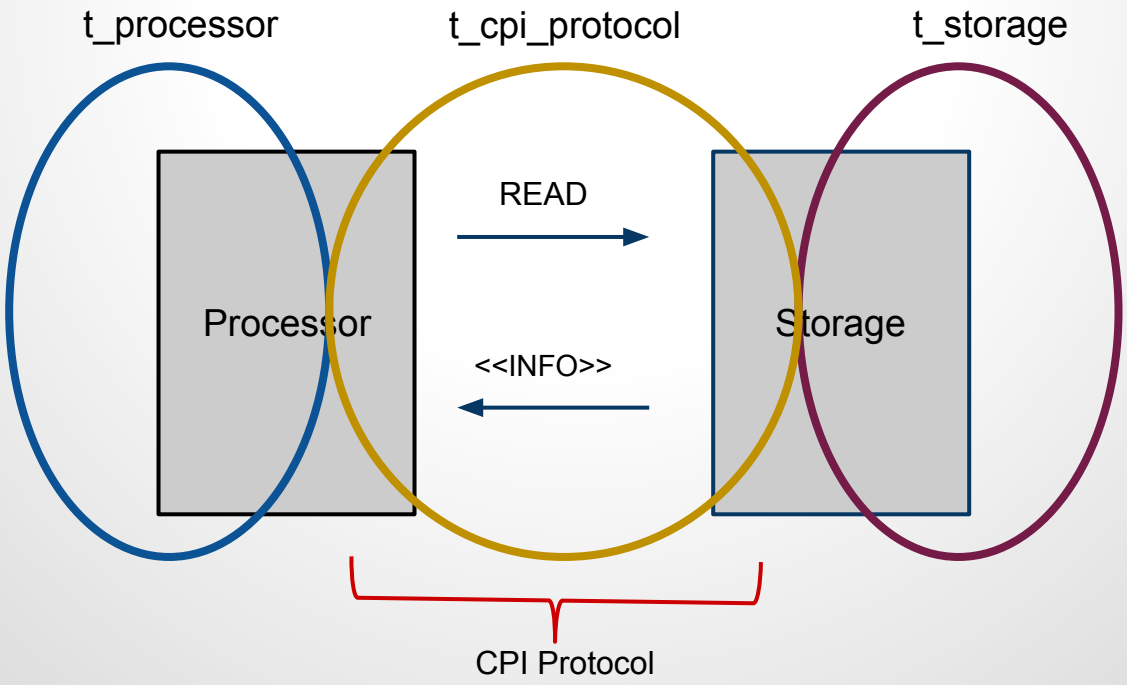


Inconvenientes:

- El Processor es desarrollado por 1 persona y el Storage por otra. Como logro coordinar el desarrollo el protocolo?
- Yo quiero asegurarme de que el protocolo funcione correctamente.
- El Processor resulta ser una tarea muy larga, mientras que el Storage es menor.
- El Storage al ser una tarea mas pequeña, va a terminar antes por lo cual necesita ser probada antes.
- Es posible que a medida que se va realizando el desarrollo del Processor, sea necesario probar algunas funcionalidades que requieren comunicación con el Storage. Comunicación que aun no se encuentra realizada.

Solución

Comprendemos que por cada proceso existe un TAD que lo representa (t_processor y t_storage). Pero consideramos al protocolo CPI como un TAD independiente (Es decir una libreria que es utilizada por los otros 2 TADs). A esto lo podemos llamar interfaz, ya que actúa como una medio de comunicación entre los 2 TADs abstrayéndonos de la implementación.



Beneficios:

- Ya no tengo 2 tareas, sino tengo 3 y puedo delegar esta 3ra a alguien mas o a alguno de los que ya esta haciendo una de las otras tareas sin necesidad de tocar el código del otro.

- Es muy facil asegurar el funcionamiento del protocolo, porque ahora es una librería independiente la cual puedo testear.

- Tanto t_processor y t_storage incluyen t_cpi_protocol y utilizan sus funciones aunque estas no estén implementadas. Esto nos permite seguir codificando sin necesidad de esperar una implementación inmediata.

- En caso de necesitar realizar pruebas podemos definir nuestra propia implementación de t_cpi_protocol la cual simule respuestas.

cpi_protocol.h

```
 *      Author: facundo
 */

#ifndef CPI_PROTOCOL_H_
#define CPI_PROTOCOL_H_

    typedef enum {
        CPI_READ            = 0x0,
        CPI_WRITE           = 0x21,
        CPI_DELETE          = 0x22,
        CPI_UPDATE          = 0x23
    }e_cpi_pkg_type;

    typedef enum {
        CPI_STORAGE         = 0X1,
        CPI_PROCESSOR       = 0X2
    }e_cpi_mode;

    typedef struct {
        int desc;
        struct sockaddr_in* my_addr;
    } t_cpi_connection;

    typedef struct {
        unsigned char           action;
        unsigned int            entry_id;
        void                    *data;
    } __attribute__((__packed__)) t_cpi_pkg;

    t_cpi_connection    *cpi_protocol_create(e_cpi_mode mode, const char* ip, int port);
    t_cpi_pkg           *cpi_protocol_read(t_cpi_connection *connection, unsigned int entry_id);
    int                 cpi_protocol_write(t_cpi_connection *connection, unsigned int entry_id, char data[]);
    int                 cpi_protocol_close(t_cpi_connection *connection);

#endif /* CPI_PROTOCOL_H_ */
```

[Esto no es un ejemplo completo]

Bibliografía utilizada

- The C Programming Language 2nd Edition - Kernighan & Ritchie
- Linux Programming Language Unleashed - Kurtwall
- Tipos Abstractos de Datos: Módulo IV K1GT5 - Cátedra SSL (Muchnik-Sola)
- Notes on Coding Style for C Programming (from K&R Style)**:** **http://www.cas.mcmaster.ca/~carette/SE3M04/2004/slides/CCodingStyle.html**
- TAD: Clases, TPs, enunciados y parciales de cursadas 2007.1C y 2008.1C de Sintaxis y Semánticas de los Lenguajes, UTN-FRBA. Prof. Ing. José María Sola. http://groups.yahoo.com/group/UTNFRBASSL/ y Fotocopiadora UTN-FRBA.