Programmation cartes à puces

Loïc Demange
loic.demange@etud.univ-paris8.fr

avec les notes de cours de Philippe Guillot

14/18 février 2021

Transactions ACID

Que se passe-t-il s'il y a une coupure électrique au moment de l'écriture des données ?
Que se passe-t-il si on arrive à rendre la taille des données invalide ?

De manière générale, comment faire en sorte que les données de la carte soient intègres ?

Transactions ACID

Réponse : les transactions, avec l'aide de la propriété **ACID**.

- **A** : Atomicité, une transaction est atomique, non divisible.
- **C** : Cohérence, l'état du système doit être cohérent.
- **I** : Isolation, chaque transaction est indépendante du reste du système.
- **D** : Durable, si la transaction réussit il faut que le système soit dans un état durable ("échec" ou "réussite").

Concrètement, soit une opération se déroule entièrement, soit pas du tout.

Transactions ACID

Une façon de faire est de le décomposer en deux étapes.

- L'engagement, qui permet d'écrire la taille, les données à écrire ainsi que leurs destinations dans une structure (elle-même dans l'EEPROM).
- La validation, qui permet d'écrire le contenu de la structure dans la mémoire actuelle.

Et deux états (eux aussi dans l'EEPROM).

- État vide : la structure de l'engagement est vide, ou non finalisée.
- État plein : la structure de l'engagement contient des données, et est finalisée.

Transactions ACID

- L'engagement ne passe à l'état plein que lorsqu'il a entièrement fini de remplir la structure.
- La validation ne passe à l'état vide que lorsqu'elle a entièrement fini d'écrire les données.

Si l'engagement échoue, on perd l'opération.
Si la validation échoue, on recommence jusqu'à succès.

Transactions ACID

```
#include <stdarg.h>
// Transaction à états
typedef enum{vide = 0, plein = 0x1c} etat_struct;

// Représente l'état de la structure
etat_struct ee_etat EEMEM = vide;

// Structure en EEPROM contenant les éléments à valider dans la carte
struct donnees {
    uint8_t nb_ope; // nombre d'opérations à réaliser
    uint8_t tailleTotale; // taille totale des données dans le buffer
    uint8_t n[NBROPE]; // taille de chacune des opérations
    uint8_t * d[NBROPE]; // destination de chacune des opérations
    uint8_t buffer[TAILLEMAX]; // buffer contenant toutes les données
} ee_aEcrire EEMEM;

// Fonctions
void engager_donnees(int n1, ...); // voir la structure va_list
void valider();
```

Projet porte monnaie

Implémenter les transactions dans votre porte monnaie.