Types opaques

Programmation séquentielle en C, 2021-2022

Orestis Malaspinas (A401), ISC, HEPIA

2022-01-18

Inspirés des slides de F. Glück

Types composés

- Jusqu'ici les `struct` sont dans les `.h` et sont *transparents*

```c
// table.h
typedef struct _table {
    int *data;
    int length;
} table;
// main.c
table tab; // membres de tab accessibles directement
tab.length = 10;
tab.data = malloc(tab.length * sizeof(int));
tab.data[9] = 10;
```

Types opaques

- Afin de cacher les détails de l'implémentation.
- Afin d'éviter les modifications directs des données.
- Afin de protéger le monde de la dévastation!
- Définition de types **opaques**:
  - Variables dans les structures ne sont pas accessibles.
  - Variables dans les structures ne sont pas modifiables.
  - Les variables ne sont même pas connues.
- Nécessité de passer par des fonctions pour initialiser/modifier les instances de types opaques.
- Très souvent utilisés pour les structures de données abstraites (table de hachage, pile, file, . . .).

Utilisation d'un type opaque: problème?

- Dans `opaque.h`

```
struct table;
```

- Dans `opaque.c`

```
struct table {
    int a;
}
```

- Dans `main.c`

```
int main() {
    struct table t;
}
// error: storage size of 't' isn't known
```

- La taille de `table` n'est pas connue à la compilation!

Utilisation d'un type opaque: pointeur!

- Dans opaque.h

```
struct table;
struct table *create();
void init(struct table **t);
```

- Dans opaque.c

```
struct table {
    int a;
}
struct table *create() {
    struct table *t = malloc(sizeof(*t));
    return t;
}
void init(struct table **t) {
    *t = malloc(sizeof(**t));
```