Pointeurs avancés

Programmation séquentielle en C, 2024-2025

Orestis Malaspinas (A401)

2025-03-21

Informatique et Systèmes de Communication, HEPIA

Pointeurs et `const`

- Le mot-clé `const` permet de déclarer des valeurs **constantes** qui ne changeront plus en cours d'exécution du programme.

```
const int a = 1;
a = 2; // interdit, erreur de compilation!
```

Deux niveaux de constance

- Mais qu'est-ce que cela veut dire pour les pointeurs?
- Constance de la valeur de l'adresse? de la valeur pointée? des deux?

```
int n = 12;
const int *p = &n; // la valeur *p est const, p non
int const *p = &n; // la valeur *p est const, p non
int *const p = &n; // la valeur p est const, *p non
const int *const p = &n; // la valeur p et *p sont const
```

Pointeurs et const

Exemples

```
int n = 12; int m = 13;
const int *p = &n; // la valeur *p est const, p non
*p = m; // erreur de compilation.
p = &m; // OK
int const *p = &n; // la valeur *p est const, p non
*p = m; // erreur de compilation.
p = &m; // OK
int *const p = &n; // la valeur p est const, *p non
*p = m; // OK
p = &m; // erreur de compilation.
const int *const p = &n; // la valeur p et *p sont const
*p = m; // erreur de compilation.
p = &m; // erreur de compilation.
```

Rappel: pointeurs et fonction

Faites un dessin de ce qui se passe en mémoire

```c
void foo(int *a) {
    *a = 3;
}
void bar(int a) {
    a = 12;
}
int main() {
    int a = 1;
    foo(&a); // Que vaut a?
    bar(a);  // Que vaut a?
}
```

Pointeurs et `const`

Fonctions

```
void foo(int *a);
void foo(const int *a); // on pourra pas changer *a
void foo(int *const a); // inutile on peut pas changer a
void foo(const int *const a); // identique à ci-dessus
```

Mais.....

```
const int a = 0;
int *b = (int *)&a;
*b = 7;
printf("a = %d\n", a); // affiche quoi?
```

Utilité

- Permet d'empêcher une mauvaise utilisation des arguments,
- Permet de documenter le code: on sait que la variable ne sera pas modifiée.