Pointeurs en C

Hicham Janati

1 Tutoriel

Un pointeur est une variable contenant l'adresse d'une autre variable. Ils nous permettent de manipuler les variables en passant par leur adresse (on verra plus tard pourquoi c'est plus puissant).

RAPPEL : Supposons deux variables a et b déclarées. On peut accéder à l'adresse de a avec le caractère "&". Voici un schéma montrant un pointeur p "pointant" sur la variable a :

Adresse	...	&a	...	&p	...	...
Contenu	...	a	...	&a	...	...

Pour déclarer un pointeur, il faut préciser le type sur lequel il va pointer suivi d'un astérisque. Pour afficher un pointeur, on utilise le format "%p" :

1. **Reprendre ce code sur votre machine :**

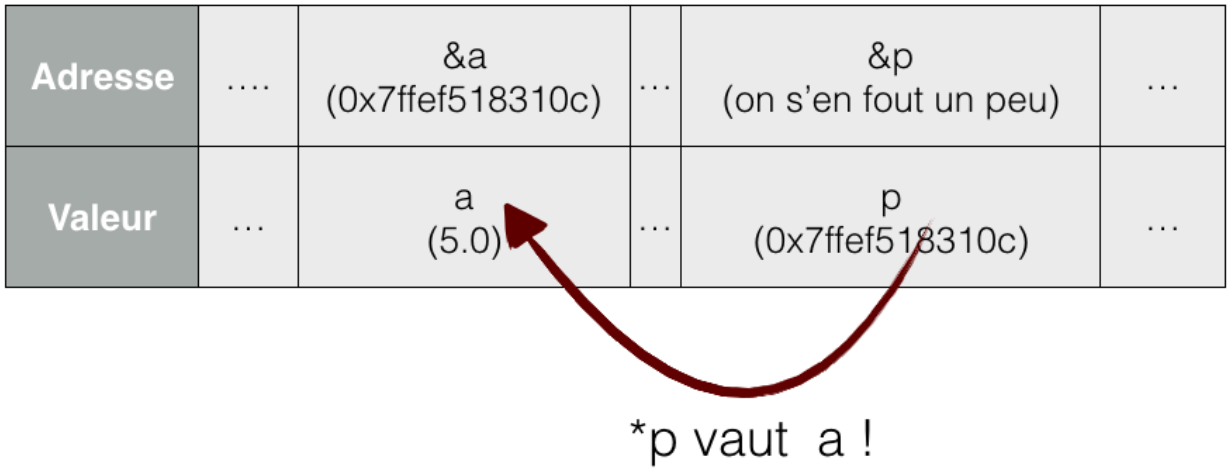
```c
#include<stdio.h>
int main(){
   float a = 5;
   float* p = &a;

   // On affiche la valeur de a et son adresse:
   printf("La variable a = %f\n",a);
   printf("L'adresse de a = %p\n",&a);

       // La valeur de p et son adresse
   printf("La valeur de p = %p\n", p);
   printf("L'adresse de p = %p\n", &p);

   return 0;
}
```

```
sh-4.2$ main
La variable a = 5.000000
L'adresse de a = 0x7ffef518310c
La valeur de p = 0x7ffef518310c
L'adresse de p = 0x7ffef5183100
```

On remarque bien que la valeur de p est bien égale à l'adresse de a ! Cool mais ça sert à quoi au juste ?
Eh ben en fait on peut *accéder à la VALEUR de a* en passant par p : il suffit d'ajouter un * avant le pointeur :

```c
   printf("La valeur de  *p = %f\n", *p);
```

```
La valeur de  *p = 5.000000
```



À RETENIR : Si p est un pointeur sur a :

- p vaut l'adresse de a : &a.
- *p vaut la valeur de a : a.
- pour afficher une adresse (un pointeur) on utilise le format %p.
- pour déclarer un pointeur, il faut préciser le type sur lequel il va pointer suivi d'un *.

2. **Créez un pointeur sur un entier et changez sa valeur en passant par le pointeur.**

3. **Reprenez cette fonction et devinez pourquoi la ligne 8 n'a aucun effet !**

```
1  #include<stdio.h>
2  void Changer(int a){
3          a = 0;
4      }
5  int main(){
6      int a = 10;
7      printf("La variable a = %d\n",a);
8      Changer(a);
9      printf("La variable a = %d\n",a);
10     return 0;
11 }
```

```
1      La variable a = 10
2      La variable a = 10
```

En passant la variable a à la fonction *Changer*, la fonction n'opère pas *directement* sur la variable a en mémoire mais crée une copie de a ! à cette copie, elle affecte la valeur 0. Mais comme la copie est une variable *locale* à la fonction, elle est détruite à la sortie de la fonction. La variable a reste inchangée.

4. **Pour y remédier, au lieu de passer à la fonction la variable, on lui donne un pointeur sur a : elle aura ainsi un accès direct à sa place en mémoire et pourra changer sa valeur. Remplacez la fonction Changer par la suivante, quel changement doit être effectué à main pour utiliser cette nouvelle fonction ?**

```
1  #include<stdio.h>
2  void Changer(int* a){
3          *a = 0;
4      }
```