Pointeurs en C.

Définition d'un pointeur

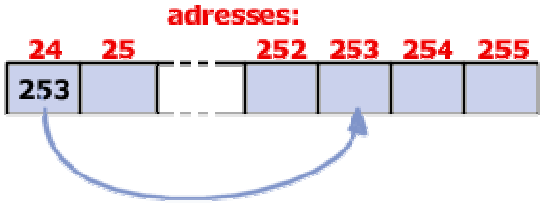
Un pointeur est une variable contenant l'adresse d'une autre variable d'un type donné. La notion de pointeur fait souvent peur car il s'agit d'une technique de programmation très puissante, permettant de définir des structures dynamiques, c'est-à-dire qui évoluent au cours du temps (par opposition aux tableaux par exemple qui sont des structures de données statiques, dont la taille est figée à la définition).

Comprendre la notion d'adresse

Comme nous l'avons vu, un pointeur est une variable qui permet de stocker une adresse, il est donc nécessaire de comprendre ce qu'est une adresse.
Lorsque l'on exécute un programme, celui-ci est stocké en mémoire, cela signifie que d'une part le code à exécuter est stocké, mais aussi que chaque variable que l'on a défini a une zone de mémoire qui lui est réservée, et la taille de cette zone correspond au type de variable que l'on a déclaré.
En réalité la mémoire est constituée de plein de petites cases de 8 bits (un octet). Une variable, selon son type (donc sa taille), va ainsi occuper une ou plusieurs de ces cases (une variable de type char occupera une seule case, tandis qu'une variable de type long occupera 4 cases consécutives).
Chacune de ces « cases » (appelées **blocs**) est identifiée par un numéro. Ce numéro s'appelle **adresse**.
On peut donc accéder à une variable de 2 façons :

- grâce à son nom
- grâce à l'adresse du premier bloc alloué à la variable

Il suffit donc de stocker l'adresse de la variable dans un pointeur (il est prévu pour cela) afin de pouvoir accéder à celle-ci (on dit que l'on « pointe vers la variable »).



Le schéma ci-dessus montre par exemple par quel mécanisme il est possible de faire pointer une variable (de type *pointeur*) vers une autre. Ici le pointeur stocké à l'adresse 24 pointe vers une variable stockée à l'adresse 253 (les valeurs sont bien évidemment arbitraires).

Comment connaît-on l'adresse d'une variable ?

En réalité vous n'aurez jamais à écrire l'adresse d'une variable, d'autant plus qu'elle change à chaque lancement de programme étant donné que le système d'exploitation alloue les blocs de mémoire qui sont libres, et ceux-ci ne sont pas les mêmes à chaque exécution.
Ainsi, il existe une syntaxe permettant de connaître l'adresse d'une variable, connaissant son nom :
il suffit de faire précéder le nom de la variable par le caractère *&* (« ET commercial ») pour désigner l'adresse de cette variable :

```
&Nom_de_la_variable
```

Intérêt des pointeurs

Les pointeurs ont un grand nombre d'intérêts :

- Ils permettent de manipuler de façon simple des données pouvant être importantes (au lieu de passer à une fonction un élément très grand (en taille) on pourra par exemple lui fournir un pointeur vers cet élément...)
- Les tableaux ne permettent de stocker qu'un nombre fixé d'éléments de même type. En stockant des pointeurs dans les cases d'un tableau, il sera possible de stocker des éléments de taille diverse, et même de rajouter des éléments au tableau en cours d'utilisation (c'est la notion de tableau dynamique qui est très étroitement liée à celle de pointeur)

- Il est possible de créer des structures chaînées, c'est-à-dire comportant des maillons

Déclaration d'un pointeur

Un pointeur est une variable qui doit être définie en précisant le type de variable pointée, de la façon suivante :
`type * Nom_du_pointeur`
Le type de variable pointée peut être aussi bien un type primaire (tel que *int*, *char*...) qu'un type complexe (tel que *struct*...).
*Un pointeur doit **préférentiellement** être typé !*
*Il est toutefois possible de définir un pointeur sur 'void', c'est-à-dire sur quelque chose qui n'a pas de type prédéfini (void * toto). Ce genre de pointeur sert généralement de pointeur de transition, dans une fonction générique, avant un transtypage permettant d'accéder effectivement aux données pointées.*
Grâce au symbole '*' le compilateur sait qu'il s'agit d'une variable de type *pointeur* et non d'une variable ordinaire, de plus, étant donné que vous précisez (obligatoirement) le type de variable, le compilateur saura combien de blocs suivent le bloc situé à l'adresse pointée.

Initialisation d'un pointeur

Après avoir déclaré un pointeur il faut l'initialiser. Cette démarche est très importante car lorsque vous déclarez un pointeur, celui-ci contient ce que la case où il est stocké contenait avant, c'est-à-dire n'importe quel nombre. Autrement dit, si vous n'initialisez pas votre pointeur, celui-ci risque de pointer vers une zone hasardeuse de votre mémoire, ce qui peut être un morceau de votre programme ou... de votre système d'exploitation !
Un pointeur non initialisé représente un danger !
Pour initialiser un pointeur, il faut utiliser l'opérateur d'affectation '=' suivi de l'opérateur d'adresse '&' auquel est accollé un nom de variable (celle-ci doit bien sûr avoir été définie avant...) :
`Nom_du_pointeur = &nom_de_la_variable_pointee;`
Par exemple :

```
int a = 2;

char b;

int *p1;

char *p2;

p1 = &a;

p2 = &b;
```

Accéder à une variable pointée

Après (et seulement après) avoir déclaré et initialisé un pointeur, il est possible d'accéder au contenu de l'adresse mémoire pointée par le pointeur grâce à l'opérateur '*'. La syntaxe est la suivante :
Par exemple :

```
int a = 2;

*p1 = 10;

*p2 = 'a';
```

Après ces deux instructions, le contenu des variables a et b sera respectivement 10 et 97 (61 en hexadécimal, le code ASCII associé au caractère 'a').
Si vous désirez utiliser cette notation dans une expression plus complexe, il sera nécessaire d'employer des parenthèses :
Par exemple :

```
a = (*p)++;
```