Quelle est l'utilité de realloc ?

realloc() permet de changer la taille du segment mémoire allouée avec malloc() pointée par un pointeur. Par exemple, pour un tableau pTab de dix entiers, grâce à realloc, vous pouvez faire qu'il contienne vingt entiers :

```
int* pTab = malloc(10 * sizeof(int)) ;
int* pBigTab = realloc(pTab, 20 * sizeof(int)) ;
if(pBigTab)
        free pBigTab ;
else
        free pTab ;
```

Déjà, on remarque que le code est plus compliqué que prévu.

Comment (bien) utiliser realloc ?

Comme on peut le voir dans l'entrée de FAQ précédente, realloc possède de nombreux cas à gérer. Énumérons-les :

- si la taille passée à realloc est de 0, realloc() est équivalent à free() ;
- si le pointeur passé à realloc est NULL, realloc() est équivalent à malloc() ;
- si realloc() ne peut allouer de la mémoire, il retourne NULL et laisse le pointeur d'origine intouché (non libéré, non modifié) ;
- autrement, realloc alloue une nouvelle zone mémoire, de la taille souhaitée, y copie N octets (le minimum entre la taille du pointeur passé et la taille donnée à réalloc) provenant du pointeur d'origine et libère le tableau d'origine.

Une particularité de realloc() est que le pointeur retourné peut être équivalent au pointeur passé et c'est de cette particularité que des erreurs de programmation en découle. Il faut bien comprendre ici, que realloc peut (mais n'est pas obligé) renvoyé le pointeur passé.
Surtout, il faut bien gérer tous les cas, pour éviter une quelconque erreur de segmentation (et faille de sécurité).

Reprenons le code précédent et décortiquons le :

```
int* pTab = malloc(10 * sizeof(int)) ;
int* pBigTab = realloc(pTab, 20 * sizeof(int)) ;
if(pBigTab) // S'il realloc retourne un pointeur valide (non nul), alors, la mémoire de pTab a été libéré, on peut écraser
        pTab = pBigTab; // Si jamais realloc() a réutilisé le segment mémoire pointé sur pTab, cet opération est neutre, car les deux variables ont la même valeur.
// Si realloc() a retourné NULL, nous ne passons pas dans le bloc précédent, mais pTab a été inchangé, donc nous devons libéré la mémoire pointée par pTab
free(pTab);
```

En résumé, il ne faut jamais faire :

```
pTab = realloc(pTab, 10) ;
```

Sinon, vous avez un risque de fuite de mémoire (par perte de la variable pointée sur une zone allouée).