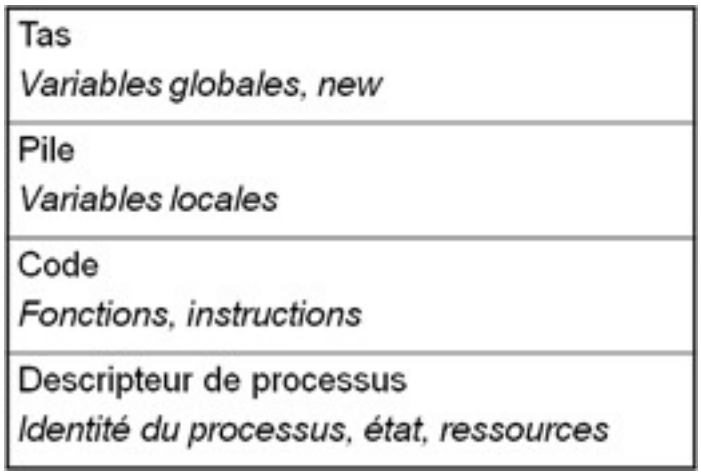
Gestion de la mémoire

La mémoire est vitale pour le système d'exploitation dont la tâche principale consiste à séparer chaque processus des autres, en lui allouant une quantité initiale de mémoire. Nous découvrons différentes stratégies d'allocation en fonction des systèmes. Les plus anciens (MS-DOS) et les plus basiques (sur microcontrôleur) adoptent une allocation par blocs d'une taille déterminée (64 ko par exemple), cette taille n'étant pas amenée à évoluer au cours du temps. D'autres systèmes ont recours à la pagination - une segmentation fine de l'espace mémoire en blocs de 4 ko -, ce qui autorise une gestion dynamique et efficace de la mémoire. Consécutivement à l'emploi de la mémoire paginée, l'adresse exprimée par un pointeur n'est jamais une adresse physique, mais logique, le microprocesseur se chargeant de la traduction.

Pour ce qui est du modèle applicatif, c'est-à-dire de la segmentation de la mémoire du processus, nous avons en général les zones suivantes :



Le langage C++ étant comme son prédécesseur très proche du système d'exploitation, il convient de bien connaître les différents types de gestion de la mémoire.

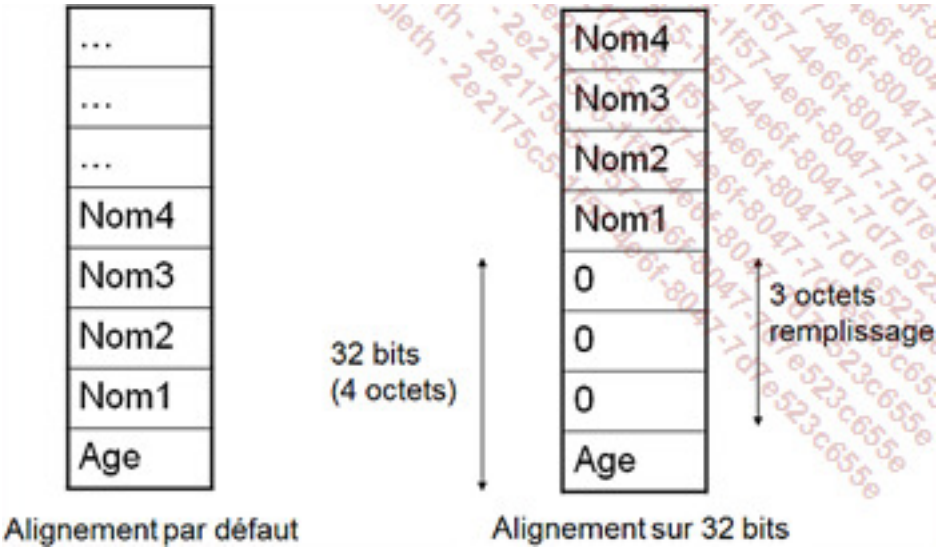
1. Alignement des données

Le développeur doit bien prendre en compte cet aspect de la gestion de la mémoire si la portabilité est un critère important. Tous les compilateurs et tous les microprocesseurs ne rangent pas les données de la même façon.

Prenons le cas de la structure `Personne` :

```
struct Personne
{
  char age;
  char*nom;
};
```

Sachant que le microprocesseur Pentium lit la mémoire par blocs de 4 octets, nombre de compilateurs insèreront 3 octets de remplissage pour aligner le champ nom sur le prochain bloc. Il en résulte une taille de structure égale à 8 alors que le comptage manuel donne 5.



Aussi, beaucoup de microprocesseurs utilisent la représentation big-endian, ce qui fait que les données sont rangées en mémoire en commençant par le poids fort. Le Pentium utilise la convention inverse. Ces caractéristiques sont déterminantes lorsqu'un fichier est échangé entre deux plates-formes, même s'il s'agit du même programme C++, compilé pour chaque plate-forme, soit avec deux compilateurs différents.

2. Allocation de mémoire interprocessus

Nous connaissons déjà deux moyens d'allouer de la mémoire dynamiquement. La fonction

`malloc()` et l'opérateur `new` ont une vocation "langage". Dans certaines situations, il est nécessaire de faire appel à d'autres fonctions offertes par le système. La mémoire partagée, utilisée par le Presse-papiers de Windows est un bon exemple.

La communication interprocessus est une prérogative du système d'exploitation. Comme beaucoup d'entre eux sont programmés en langage C ou C++, il est assez facile d'utiliser ces fonctions. Toutefois, la notion de pointeur n'existe pas dans tous les langages, et les fonctions d'allocation retournent souvent un entier, appelé **`handle`**, identifiant le bloc alloué. La désignation de cet entier prend souvent comme nom **`Hresult`** (*Handle Result*), qui est en fait un entier déguisé.

Nous retrouverons des exemples de ce type au chapitre consacré à la programmation C++ sous Windows, Les univers de C++.