La segmentation

Un·e utilisateur·ice voit un programme comme un ensemble d'instructions, de données, de fonctions, etc. Bref, un ensemble de blocs distinct dont les données ne sont pas mélangées avec les autres.

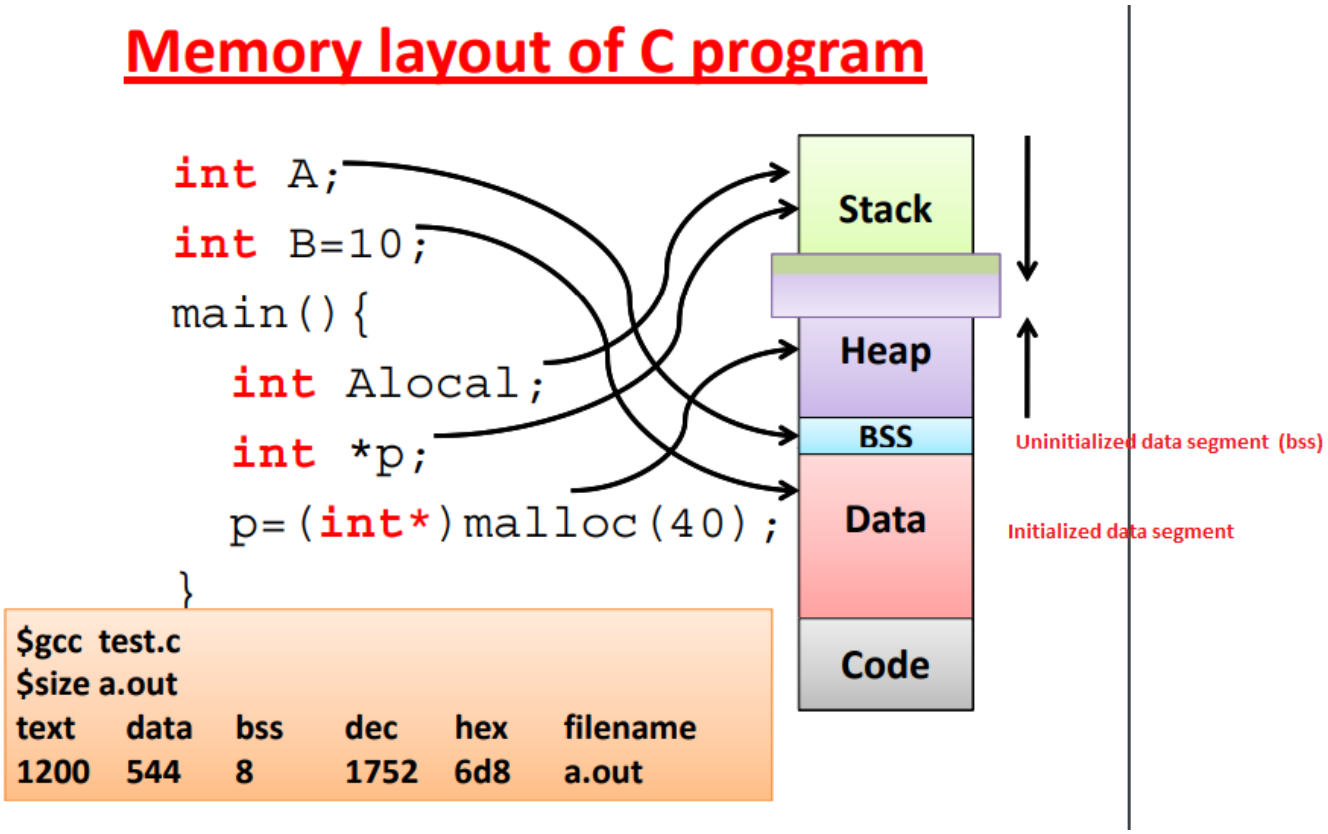
Il convient donc de traduire cette vue à l'intérieur du système d'exploitation. Pour cela, on va diviser l'espace d'adressage du processus en segments. Chaque segment a un nom, un numéro et une taille.

Les différents segments

Ainsi dans une architecture Intel x86, on peut avoir :

- Le **Code Segment (CS)** qui désigne le segment qui contient les instructions du programme (lecture seule)
- Le **Data Segment (DS)** qui contient les données du programme (variables globales, variables `static`)
- Le **Stack Segment (SS)** qui contient la pile du programme (variables locales, appels, etc)
- Le **Extra Segment (ES)** qui est un segment supplémentaire défini par le·a développeur·euse du programme

De manière plus descriptive, on peut définir la mémoire du point de vue d'un programme comme ceci. À noter cependant que, cela est surtout comme ça que fonctionnaient les anciens systèmes, mais comme on va le voir dans la suite ce n'est plus exactement comme ça que cela fonctionne.



Avec la segmentation comme ceci, une adresse logique devient ainsi `<segment-number, offset>`.

Avantages

En utilisant la segmentation, le système offre ainsi une protection adaptée car chaque segment contient des données d'une même nature et ne peuvent donc pas intervenir dans d'autres données. On ne peut pas par exemple accéder aux instructions du programme depuis le *data segment*.

Également, les segments peuvent être partagés entre plusieurs processus (par exemple avec les libraires systèmes) et ainsi faire une économie d'espace.

Désavantages

Cela fait un retour à la fragmentation, étant donné que les segments sont de taille variables, on retrouve de la segmentation externe. C'est pour cela qu'il faut combiner la segmentation avec la pagination.

Revision #1
Created 3 January 2024 14:06:17 by SnowCode
Updated 6 January 2024 19:13:15 by SnowCode