Mémoire paginée

SYE

10 - Mémoire paginée

Abstract

Ce document traite de la gestion mémoire par pagination, une technique qui segmente la mémoire en pages pour optimiser l'allocation et assurer l'isolation entre processus. Il explore les concepts d'adressage virtuel, de traduction d'adresses, et de partage de mémoire, tout en présentant des mécanismes comme les TLBs et la pagination multiniveau pour améliorer les performances. Le document aborde également les spécificités de la pagination dans différents systèmes, mettant en évidence les avantages de cette approche pour les systèmes modernes.

Table des matières

1. Introduction à la pagination

Une page mémoire est un ensemble d'octets contigus. La mémoire paginée est une technique de gestion de la mémoire qui consiste à découper la mémoire en pages de taille fixe de manière générale 4 Ko (4096 octets). Une page virtuelle est mappée sur une page physique. Les pages sont numérotées.

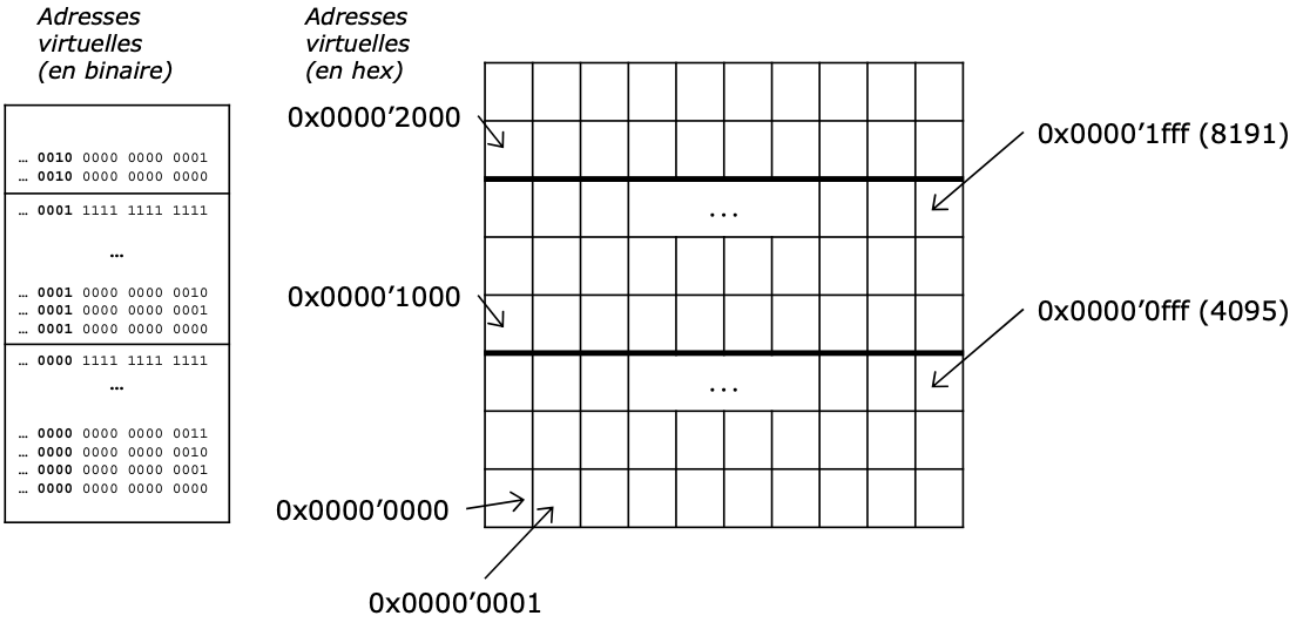
L'allocation mémoire se fait désormais par page, donc il sera impossible d'allouer moins que le taille d'une page. Allouer une page consistera donc à trouver un trou de la taille d'une page, cela permettera de résoudre presque complètement le problème de fragmentation.

1.1. Traduction d'adresse

Pour traduire une adresse virtuelle en adresse physique, on utilise une table de pages. Une adresse virtuelle est composée de deux parties : le numéro de page ainsi que le déplacement dans la page (offset). La MMU effectue la décomposition et pourra, à partir du numéro de page retrouver le numéro de page physique grace à l'utilisation de la table de pages.

1.1.1. Structure adresses virtuelles

Au niveau de l'adressage, le découpage en numéro de page et déplacement est complètement transparent, dans la mesure où il n'influe en aucune façon la taille de l'espace d'adressage, ni la signification intrinsèque de l'adresse (une adresse permet toujours de référencer un seul byte).



L'adresse est représentée sur 32 bits (pour une architecture 64 bits l'adresse serait sur 64 bits).

- le **numéro de page** correspond aux bits de poids fort de l'adresse
- le **déplacement** (offset) correspond aux bits de poids faible de l'adresse

1.2. Exemple

Soit une page faisant 4 Ko (4096 octets) et une adresse virtuelle de 32 bits. Le numéro de page sera donc sur 20 bits et le déplacement sur 12 bits car il faudra 12 bits pour représenter le déplacement à l'intérieur de la page ($2^{12} = 4096$ octets). Il reste donc 20 bits pour le numéro de page.

Avec 20 bits restant nous obtenons donc 2^{20} pages soit $1'048'576$ pages.

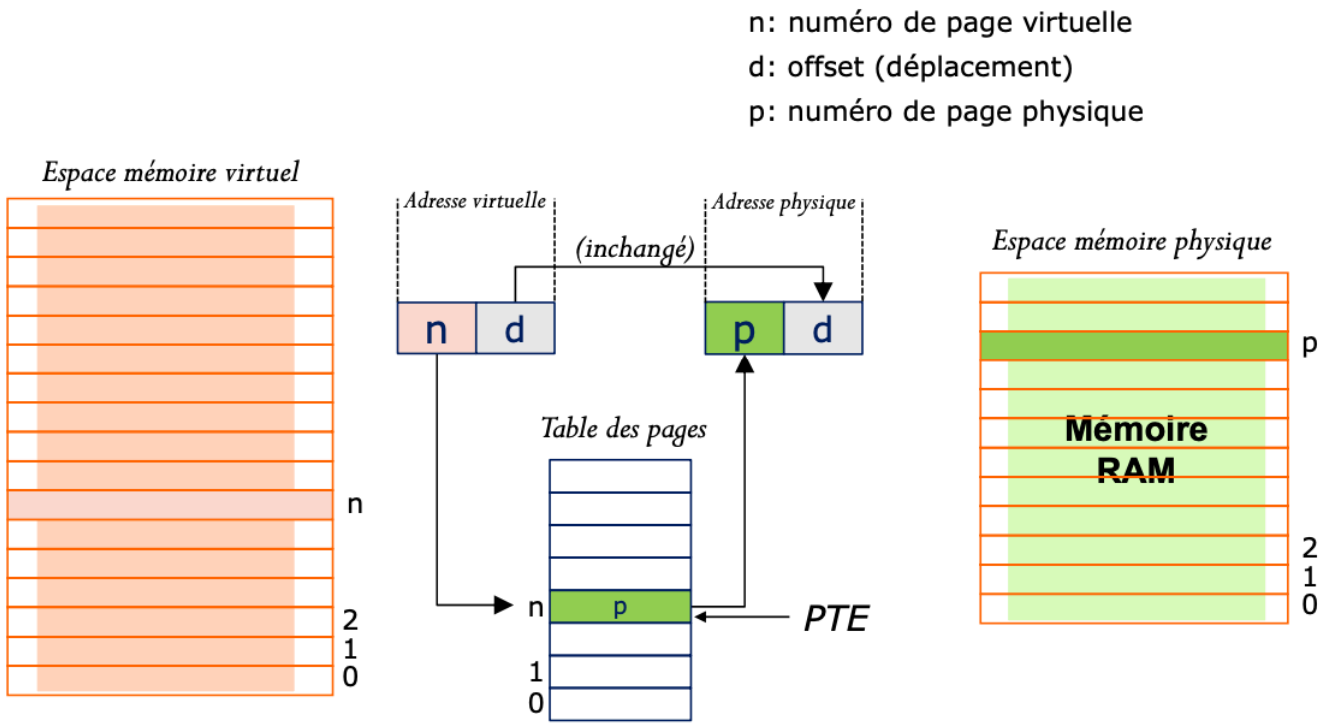
On notera que la taille d'une page dépend directement du nombre de bits représentant l'offset. Si 8 bits sont utilisés pour l'offset, la taille d'une page sera de $2^8 = 256$ octets.

2. Pagination à un niveau

Pour réaliser de la pagination à un niveau il faut utiliser une table des pages. Cette table est composée de PTE (Page Table Entry) qui contiennent l'adresse physique de la page correspondante plus un certain nombre d'informations caractéristiques de la page tel que droit d'accès, présence en mémoire, page ayant subit des modifications, etc. Habituellement une PTE fait 4 octets.

2.1. Calcul de l'adresse physique

Pour calculer l'adresse physique réelle, il faut accéder à l'entrée numéro N de la table des pages (PTE), où N correspond au numéro de la page virtuelle. Le numéro de la page physique, extrait de cette entrée, formera la partie haute (bits de poids fort) de l'adresse physique. La partie basse (bits de poids faible) sera identique à l'offset de l'adresse virtuelle. Puisque les tailles des pages sont les mêmes pour la mémoire virtuelle et la mémoire physique, l'offset reste inchangé.



3. Notion de TLB (Translation Look-aside Buffer)

Le TLB est un cache de traduction d’adresse. Il permet de stocker les traductions d’adresse les plus récemment utilisées. Lorsqu’une adresse virtuelle est traduite en adresse physique, le TLB est consulté en premier. Si la traduction est présente dans le TLB, elle est utilisée directement. Si la traduction n’est pas présente dans le TLB, la table des pages est consultée et la traduction est stockée dans le TLB pour une utilisation ultérieure.

4. Processus et pagination

4.1. Code partagé

- Certaines pages physiques peuvent être partagées entre plusieurs processus.
- Exemple : le code de la libc ou des appels système.
- Le code réentrant permet l'exécution indépendante, même en cas d'accès concurrent.

4.2. Espace d'adressage virtuel

- Un processus peut mapper des zones mémoire virtuelle sur les mêmes pages physiques qu'un autre processus.
- Les zones du noyau OS sont également mappées sur des pages communes.
- Les espaces virtuels peuvent contenir du code et des données partagés sans duplication réelle.

4.3. Tables de pages

- **Chaque processus** possède une **table des pages** en mémoire physique.
- La MMU (Memory Management Unit) gère l'accès en fonction du registre base des pages (ex. : CR3 sur Pentium, CP15:c1 sur ARM).
- À chaque changement de contexte :
  - La MMU est reconfigurée pour pointer vers la table des pages du nouveau processus.
  - Cela assure l'isolation et la gestion individuelle des espaces d'adressage.

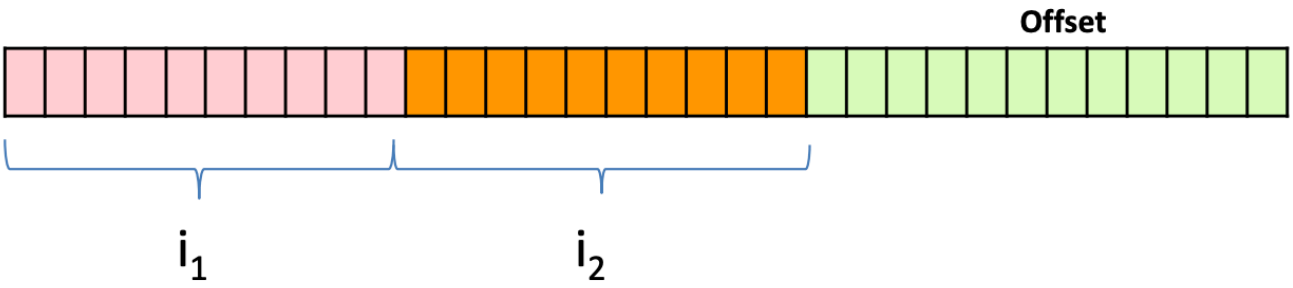
4.4. Performances et optimisation

- Les systèmes paginés nécessitent deux accès mémoire :
  1. Récupérer la PTE (Page Table Entry) dans la table des pages.
  2. Accéder aux données via l'adresse physique.
- Cela double le temps d'accès mémoire par rapport à un système sans MMU.
- Solution : des caches dédiés (TLB - Translation Lookaside Buffer) stockent temporairement les PTEs pour accélérer les accès.

5. Pagination multiniveau

La pagination à un niveau (sur une architecture 32 bits) entraîne l'utilisation de table des pages très volumineuses. Une table des pages contient autant d'entrée qu'il y a de pages virtuelles utilisable. Pour une architecture 32 bits, cela signifie 2^{20} entrées. Chaque entrée fait 4 octets, cela signifie que la table des pages fait 4 Mo que l'on doit placer en mémoire dans une zone contigue. Cela est inacceptable en terme d'utilisation mémoire.

Sur une architecture 32 bits, nous introduisons une étape supplémentaire dans le mécanisme de traduction en ajoutant une seconde table des pages. Ce mécanisme consiste à décomposer en deux parties égales les 20 bits de poids forts.



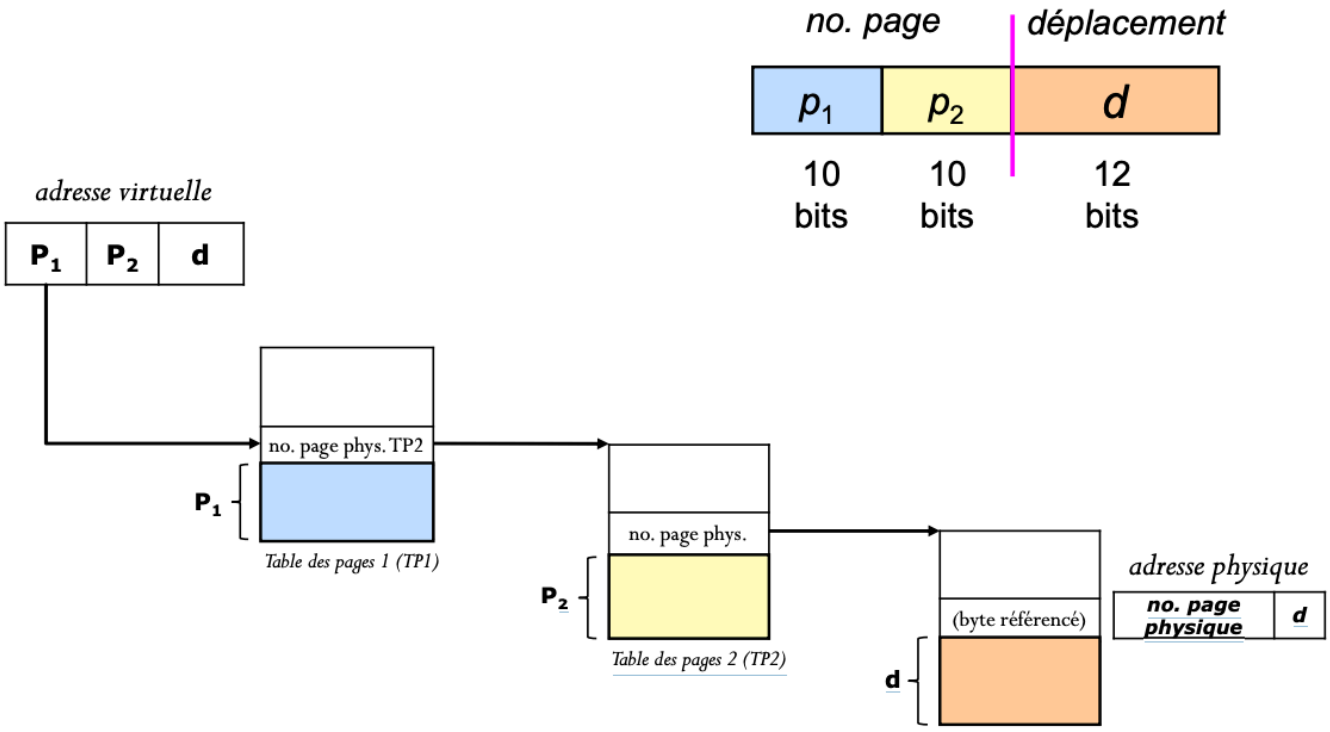
5.1. Pagination à deux niveaux

5.1.1. Taille et organisation

- Avec un index de 10 bits, chaque table peut contenir 2^{10} entrées, soit 1′024 PTEs.
- Chaque PTE étant encodée sur 4 octets, une table occupe 4′096 octets, soit la taille exacte d'une page de 4 Ko.
- Cette correspondance permet une gestion optimale : chaque table de pages utilise entièrement une page physique.

5.1.2. Fonctionnement

- Le mécanisme est similaire à une table des pages à un niveau, mais avec une indirection supplémentaire.
- La **table de premier niveau** contient des pointeurs vers des **tables de second niveau**.
- Étapes pour accéder à une donnée :
  1. Utiliser l'index de niveau 1 pour localiser la table de second niveau.
  2. Utiliser l'index de niveau 2 pour obtenir la PTE correspondante.
  3. Accéder à la page physique contenant la donnée adressée.



5.1.3. Performance et optimisation

- Cette méthode introduit un accès mémoire supplémentaire dû à l'indirection.
- Les PTEs de premier et de second niveau peuvent être stockées dans les TLBs (Translation Lookaside Buffers) pour accélérer les accès.
- Chaque processus dispose d'une table de premier niveau (1′024 entrées), où chaque entrée pointe vers une table de second niveau (1′024 entrées).
- Ce système conserve le même nombre total d'entrées qu'une table des pages à un niveau (1′024 x 1′024 = 1 million d'entrées), tout en paginant la table des pages elle-même.

5.2. Pagination sur Windows

Dans Windows, la terminologie est légèrement différente, on parle de Page Directory pour la table de premier niveau et de Page Table pour la table de second niveau. La taille de la page est de 4 Ko pour la plupart des pages mémoires. La page physique est appelée Page Frame et son numéro est appelé Physical Frame Number (PFN).

Dans les OS modernes, la pagination s'effectue à la demande (on-demand paging) : cela signifie qu'un processus au démarrage n'a aucunement besoin de charger l'ensemble de toutes ses pages, mais de se contenter de charger les pages au fur et à mesure des besoins, réduisant ainsi l'utilisation de la mémoire physique. Cette approche nécessite une gestion efficace du chargement des pages et sera décrite dans le chapitre suivant.