Démo 9

Solution

IFT2245

25 mars 2019

9.1 Fragmentation par segments

Soit un système qui utilise un système de gestion mémoire basé sur les segments.
Jouez le rôle du système d'exploitation qui doit faire fonctionner les processus suivants sur une machine avec 16MB de mémoire et donc décider où placer ces segments dans la mémoire physique, *sans savoir à l'avance quelles requêtes vont être faites*. Estimer le coût de chaque genre d'opération, et essayer de minimizer le coût total d'exécution.

9.1 Fragmentation par segments

La séquence d'événements est la suivante :

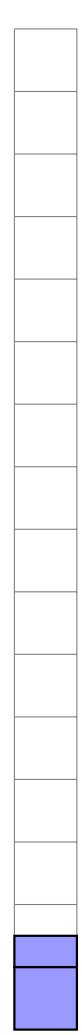
- Création de P1 avec 1MB de code et $\frac{1}{2}$MB de pile.
- Création de P2 avec 1MB de code et $\frac{1}{2}$MB de pile.
- P1 alloue un segment de 6MB.
- P1 alloue un segment de 128KB.
- P1 libère le segment de 6MB.
- P1 alloue un segment de 12MB.
- P2 alloue un segment de 128KB.
- P1 libère son segment de 12MB.
- P2 alloue un segment de 256KB.
- P1 alloue un nouveau segment de 12MB.
- P2 alloue un segment de 64KB.
- P1 libère son segment de 12MB.
- P1 alloue un segment de 7MB.
- P2 alloue un autre segment de 64KB.
- P1 termine.
- P2 alloue un segment de 128KB.
- P2 libère son segment de pile.
- P1 est relancé et ré-exécute la même séquence d'allocations/désallocations.

9.1 Fragmentation par segments

Solution

Création de P1 avec 1MB de code et $\frac{1}{2}$MB de pile.

Used memory : `1.5MB`

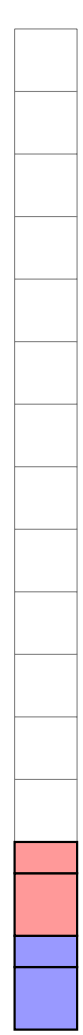


9.1 Fragmentation par segments

Solution

Création de P2 avec 1MB de code et $\frac{1}{2}$MB de pile.

Used memory : 3MB

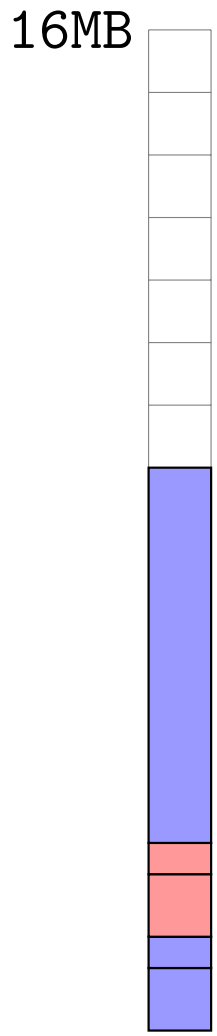


9.1 Fragmentation par segments

Solution

P1 alloue un segment de 6MB.

Used memory : 9MB

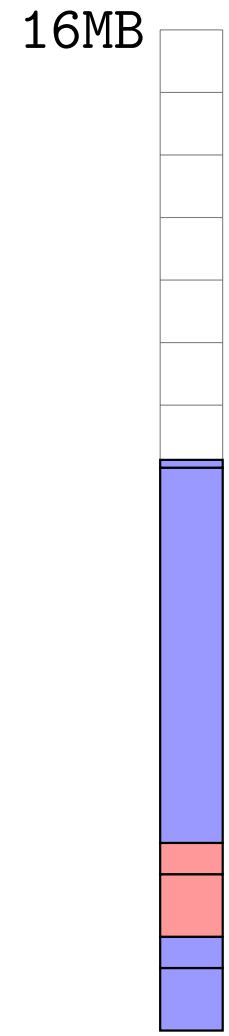


9.1 Fragmentation par segments

Solution

P1 alloue un segment de 128KB.

Used memory : `9.125MB`

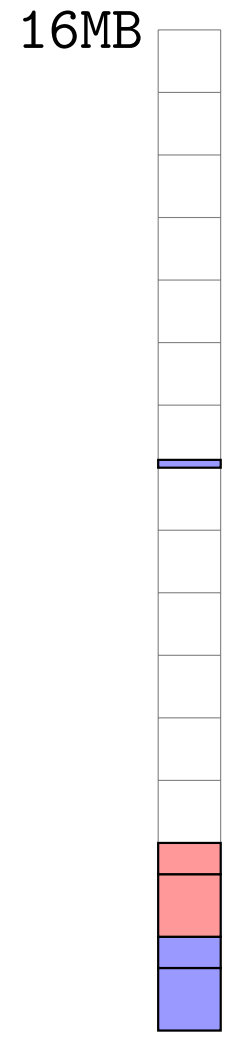


9.1 Fragmentation par segments

Solution

P1 libère le segment de 6MB.

Used memory : 3.125MB

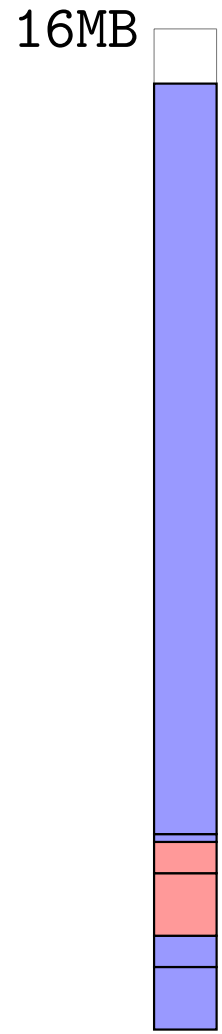


9.1 Fragmentation par segments

Solution

P1 alloue un segment de 12MB.

Used memory : `15.125MB`

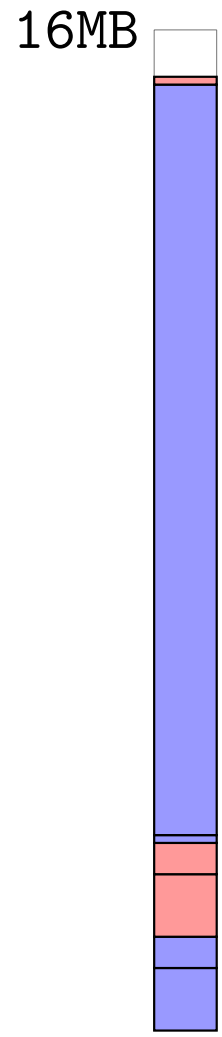


9.1 Fragmentation par segments

Solution

P2 alloue un segment de 128KB.

Used memory : `15.25MB`

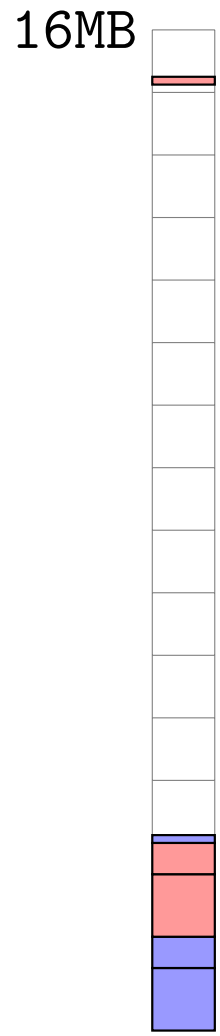


9.1 Fragmentation par segments

Solution

P1 libère son segment de 12MB.

Used memory : 3.25MB

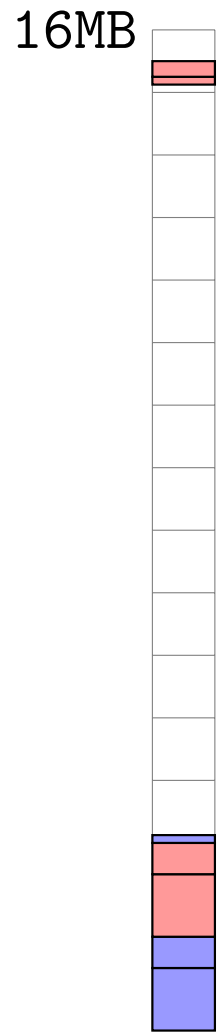


9.1 Fragmentation par segments

Solution

P2 alloue un segment de 256KB.

Used memory : 3.5MB

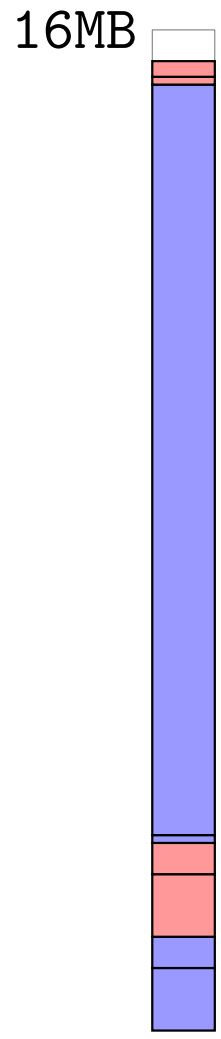


9.1 Fragmentation par segments

Solution

P1 alloue un nouveau segment de 12MB.

Used memory : `15.5MB`

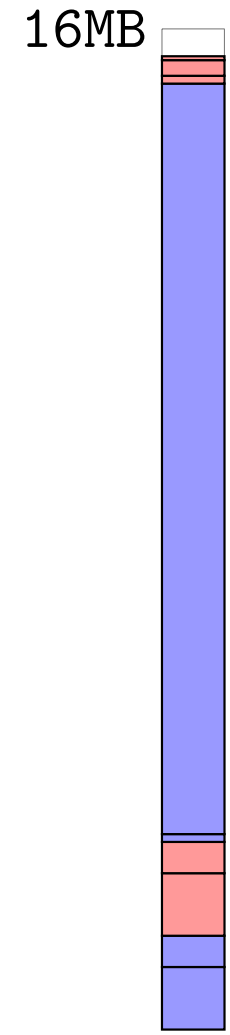


9.1 Fragmentation par segments

Solution

P2 alloue un segment de 64KB.

Used memory : `15.5625MB`

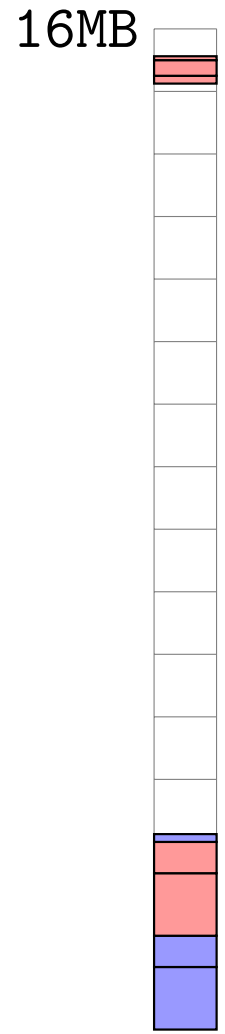


9.1 Fragmentation par segments

Solution

P1 libère son segment de 12MB.

Used memory : 3.5625MB

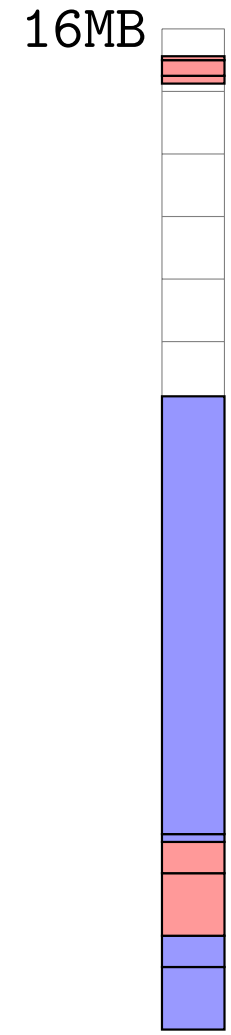


9.1 Fragmentation par segments

Solution

P1 alloue un segment de 7MB.

Used memory : `10.5625MB`

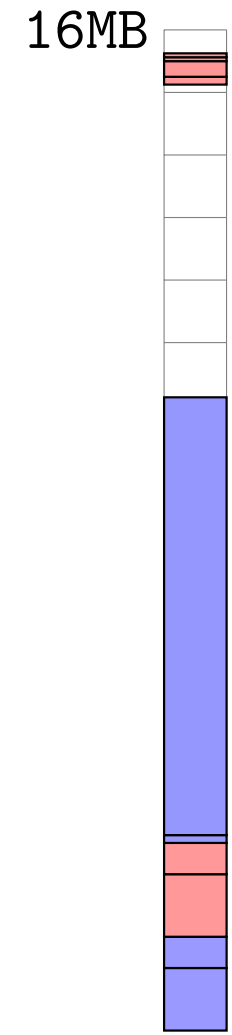


9.1 Fragmentation par segments

Solution

P2 alloue un autre segment de 64KB.

Used memory : `10.625MB`

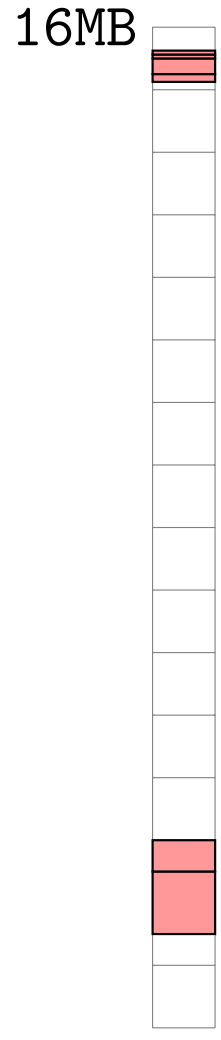


9.1 Fragmentation par segments

Solution

P1 termine.

Used memory : 2MB

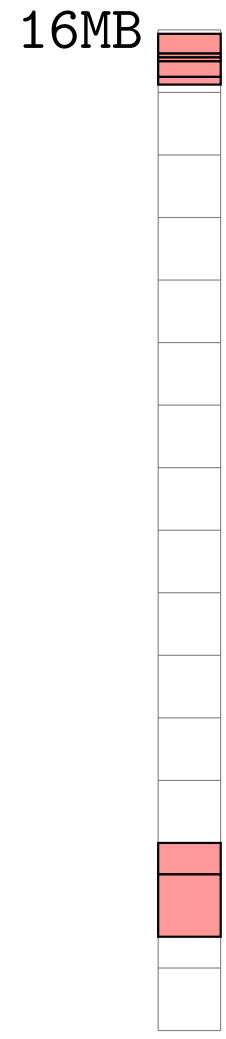


9.1 Fragmentation par segments

Solution

P2 alloue un segment de 128KB.

Used memory : 2.125MB

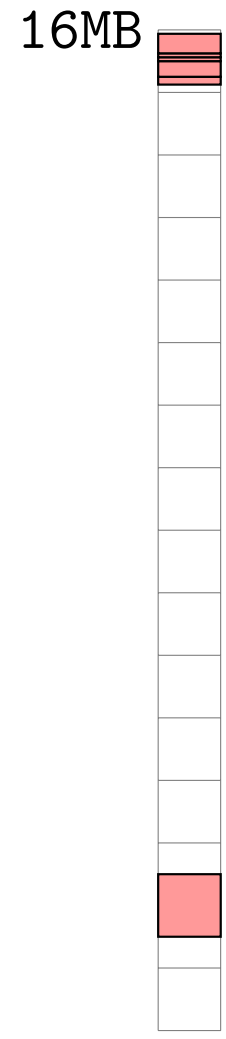


9.1 Fragmentation par segments

Solution

P2 libère son segment de pile.

Used memory : `1.625MB`



9.1 Fragmentation par segments

Solution

P1 est relancé et ré-exécute la même séquence d'allocations/désallocations.

En exercice

9.2 Fragmentation

Expliquer la différence entre la fragmentation interne et la fragmentation externe.

9.2 Fragmentation

Expliquer la différence entre la fragmentation interne et la fragmentation externe.

Solution

Fragmentation interne : Espace non utilisé à l'intérieur d'une région ou d'une page par le processus qu'elle contient. L'espace est "innaccessible" pour le système tant que le processus est en exécution.

Fragmentation externe : Espace mémoire non utilisé entre les régions de mémoire allouées. Ces trous apparaissent suite à de multiples allocations et désallocations de régions et sont généralement difficiles à remplir.

9.3 Adresses logiques

Nommer deux différence entre les adresses logiques et les adresses physiques.

9.3 Adresses logiques

Nommer deux différence entre les adresses logiques et les adresses physiques.

Solution

Les adresses physiques sont limités par la mémoire physique alors que les adresses logiques sont limités par la taille de l'espace adressable.
Les adresses logiques sont généré à l'interne par le processeur alors que les adresses physiques sont généré par le MMU.

9.4 Deux segments

Soit un système où chaque processus est divisé en 2 *segments* : un pour le code et un pour les données. I.e. il y a une paire *base+limit* utilisée pour lire les instructions et une autre pour accéder aux données. Discuter les avantages et inconvénient de ce système.

9.4 Deux segments

Soit un système où chaque processus est divisé en 2 *segments* : un pour le code et un pour les données. I.e. il y a une paire *base+limit* utilisée pour lire les instructions et une autre pour accéder aux données. Discuter les avantages et inconvénient de ce système.

Solution

Cela permet de partager le segment de code entre plusieurs instance du même programme. Permet de protéger le code contre des modification involontaire.
Si le code est partagé, cela limite la possibilité d'exécuter du code automodifiant (une pratique de moins en moins répendue). Cela oblige une séparation du code et des données.

9.5 Taille des pages

Pourquoi les pages sont-elles toujours de taille 2^n ?

9.5 Taille des pages

Pourquoi les pages sont-elles toujours de taille 2^n ?

Solution

L'entropie de Shannon. L'offset d'une page correspond à une subdivision de taille n d'une adresse. Nécessairement, l'offset obtenu permet d'adresser 2^n mot à l'intérieur de la page.

9.6 Taille des adresses

Soit un espace d'adressage logique de 64 pages de 1024 mots chacune, qui peuvent se placer dans n'importe laquelle de 32 *frames*.

1. Combien de bits y a-t-il dans une adresse logique ?
2. Combien de bits y a-t-il dans une adresse physique ?

9.6 Taille des adresses

Soit un espace d'adressage logique de 64 pages de 1024 mots chacune, qui peuvent se placer dans n'importe laquelle de 32 *frames*.

1. Combien de bits y a-t-il dans une adresse logique ?
   $64 * 1024 = 2^6 * 2^{10} = 2^{16} \rightarrow 16$
2. Combien de bits y a-t-il dans une adresse physique ?
   $32 * 1024 = 2^5 * 2^{10} = 2^{15} \rightarrow 15$

9.7 Partage de frame

Quels sont les effets de permettre à deux entrées d'une *page table* de pointer vers la même *frame* ? Expliquer comment utiliser cette fonctionnalité pour diminuer le temps nécessaire à copier une large zone de mémoire. Quel est l'effet de modifier un byte dans l'autre page ?

9.7 Partage de frame

Quels sont les effets de permettre à deux entrées d'une *page table* de pointer vers la même *frame* ? Expliquer comment utiliser cette fonctionnalité pour diminuer le temps nécessaire à copier une large zone de mémoire. Quel est l'effet de modifier un byte dans l'autre page ?

Solution

Cela permet le partage de mémoire et de code entre 2 processus.
Deux processus peuvent ainsi partager leurs pages de code.
Au lieu de copier la mémoire, on ajoute simplement une nouvelle référence au frame dans la table de pages.
Comme la mémoire est partagée, tout changement fait par l'un des processus sera visible par l'autre, ce qui n'est pas toujours désirable. Pour palier à ce problème, on introduit les mécanismes de *copy-on-write*. (fork)

9.8 Caches et pages

Un choix fondamental que doivent faire les concepteurs d'un processeur est si le cache est indexé avec des adresses logiques ou des adresses physiques.
Discuter de l'impact de ce choix sur le design du reste du processeur
(e.g. le TLB). Discuter de l'impact de ce choix sur le système d'exploitation.

9.8 Caches et pages

Un choix fondamental que doivent faire les concepteurs d'un processeur est si le cache est indexé avec des adresses logiques ou des adresses physiques.
Discuter de l'impact de ce choix sur le design du reste du processeur
(e.g. le TLB). Discuter de l'impact de ce choix sur le système d'exploitation.

Solution

Addresse physique : Nécessite la traduction des adresses virtuelles avant les caches - Latence
Addresse logique : * Le TLB peut arriver après les caches

- Meilleure latence, puisqu'on peut éviter la traduction
- Besoin d'un mécanisme avant les caches pour assuré la protection de la mémoire (peut être calculé en parallèle à l'accès aux caches) * Problème d'aliasage.
- Peut partiellement être réglé avec un flush ou un l'utilisation d'ASID (Address Space IDentifier)