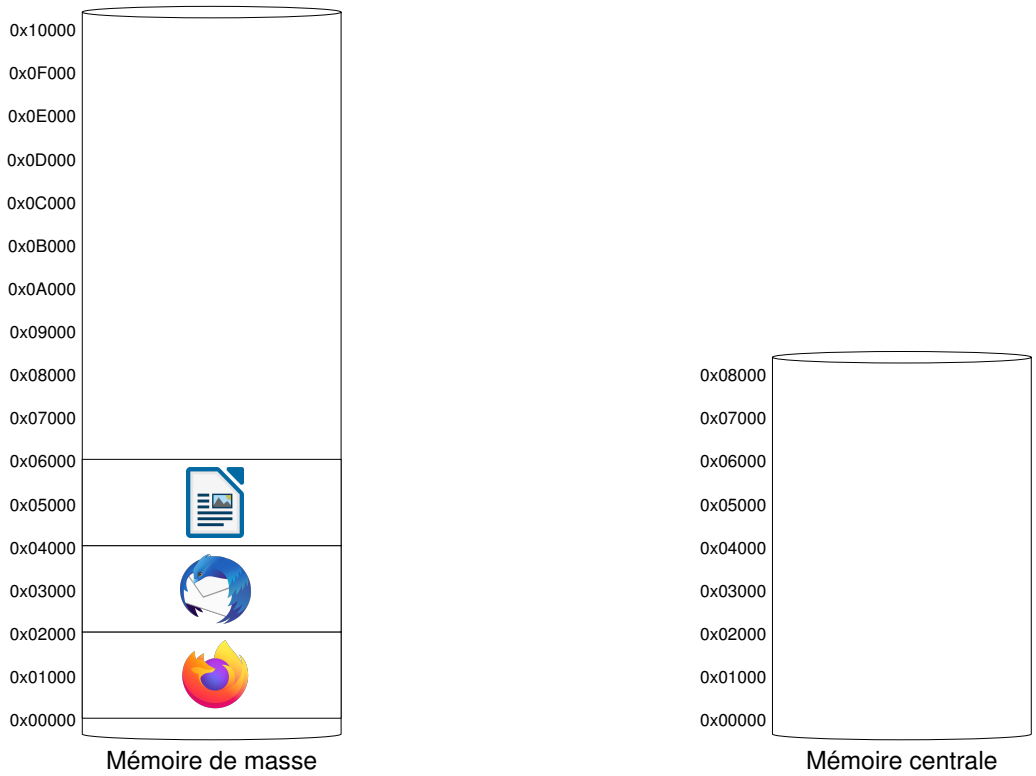
Virtualisation de la mémoire

Vincent Migliore

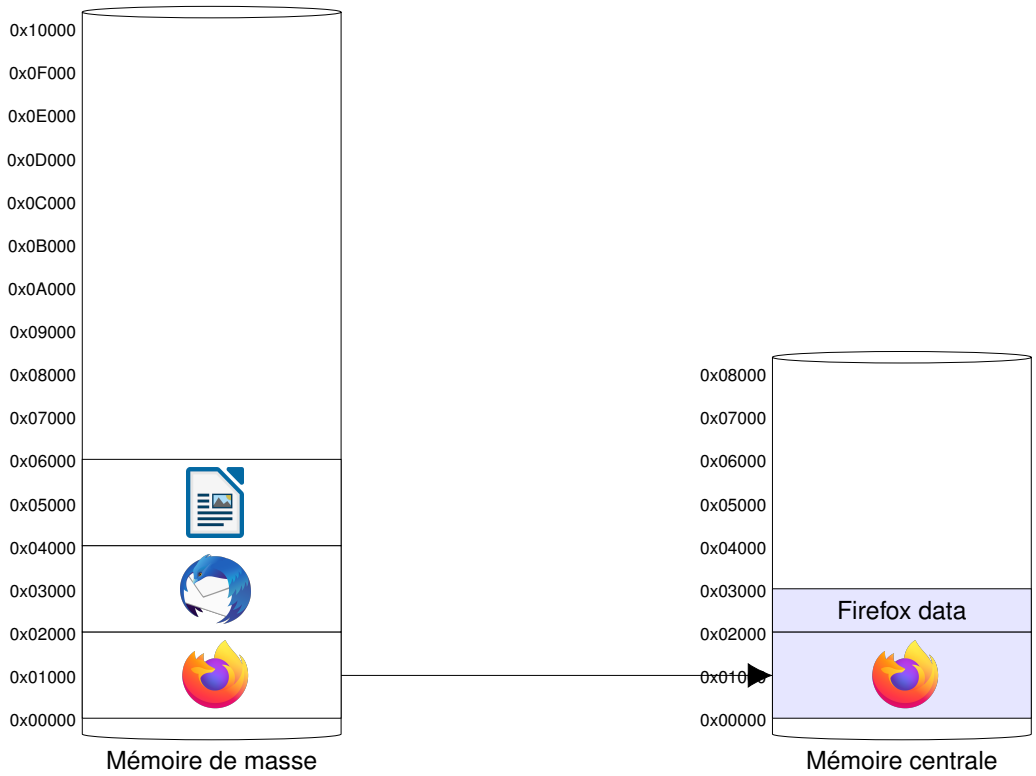
vincent.migliore@insa-toulouse.fr

Gestion de la mémoire centrale



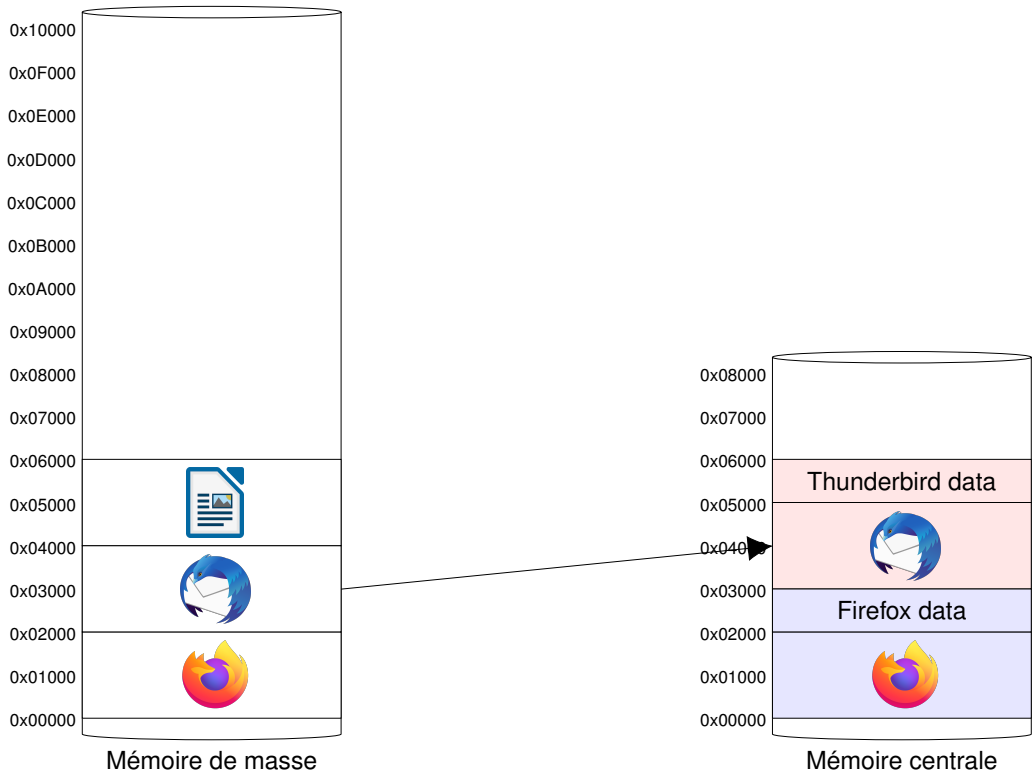
Comment est géré la mémoire centrale de l'ordinateur lorsque l'on ouvre plusieurs applications ?

Gestion de la mémoire centrale



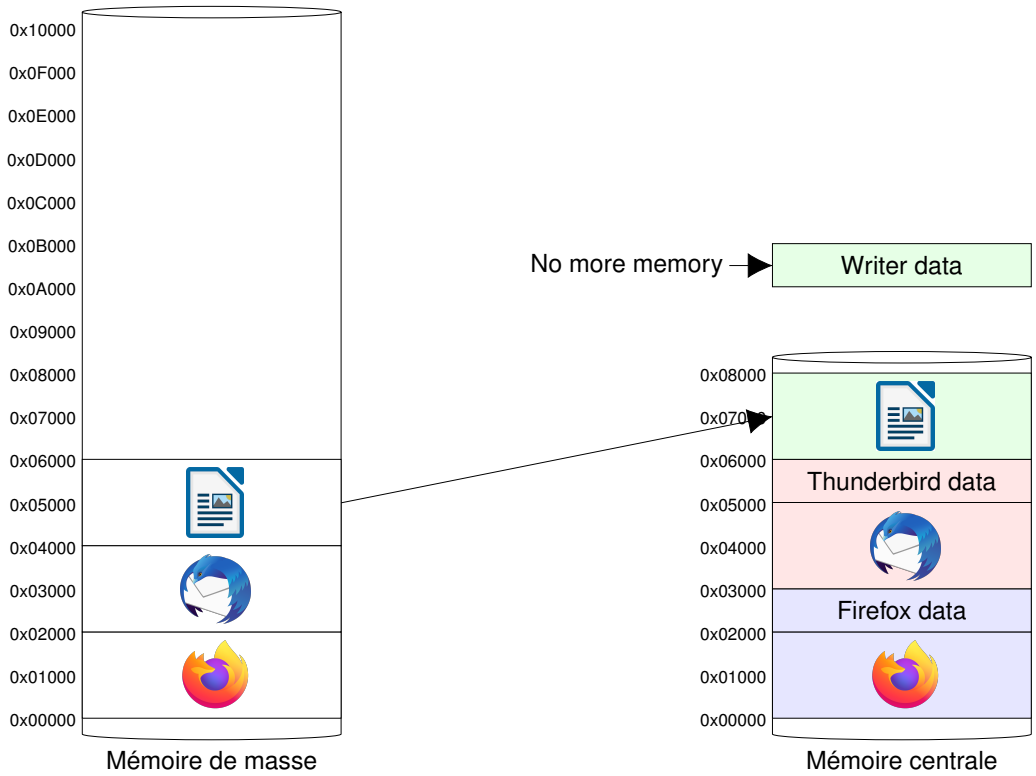
Lors de l'ouverture d'une application, son exécutable est copié dans la mémoire centrale. Un espace mémoire plus grand lui est alloué pour accueillir les données.

Gestion de la mémoire centrale



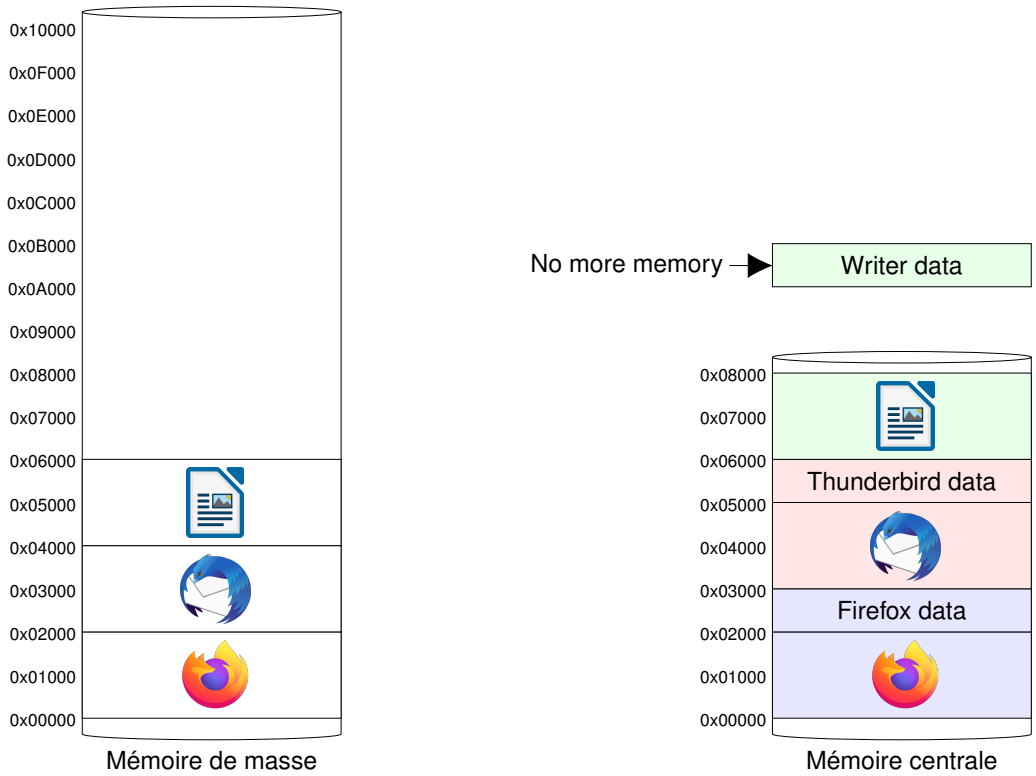
Lors de l'ouverture d'une seconde application, le système d'exploitation cherche une partie de la mémoire centrale non-utilisée, puis copie le programme et alloue de l'espace pour les données.

Gestion de la mémoire centrale

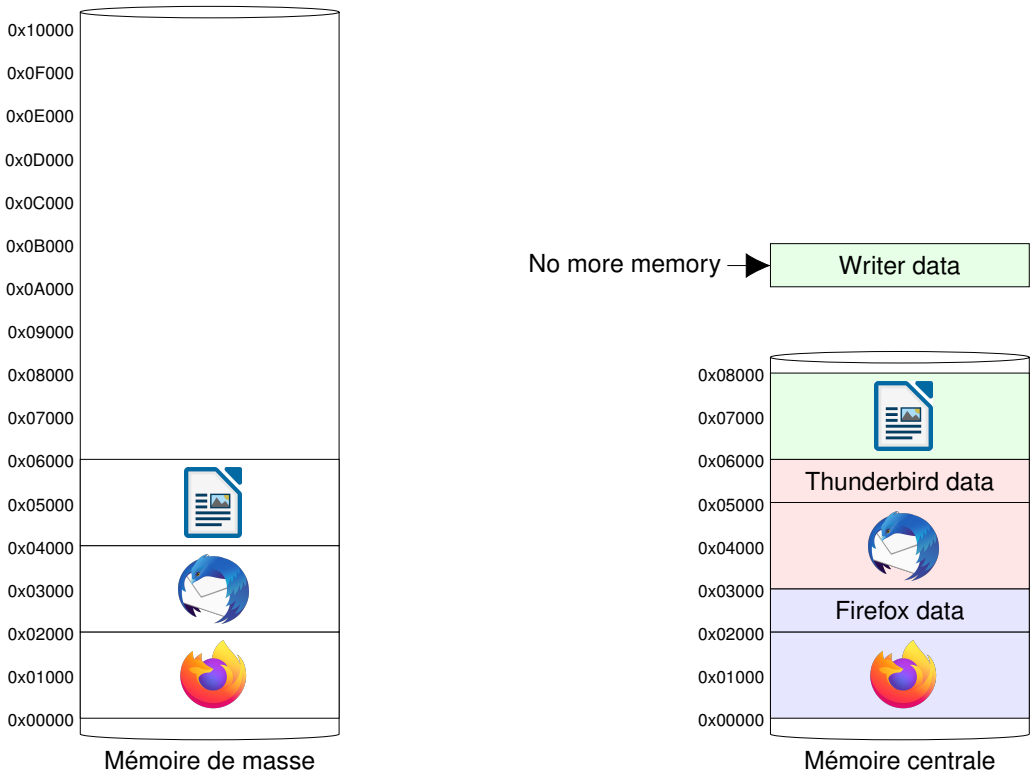


Lors de l'ouverture d'une troisième application, il n'y a plus assez d'espace disponible, le système d'exploitation doit gérer cette exception...

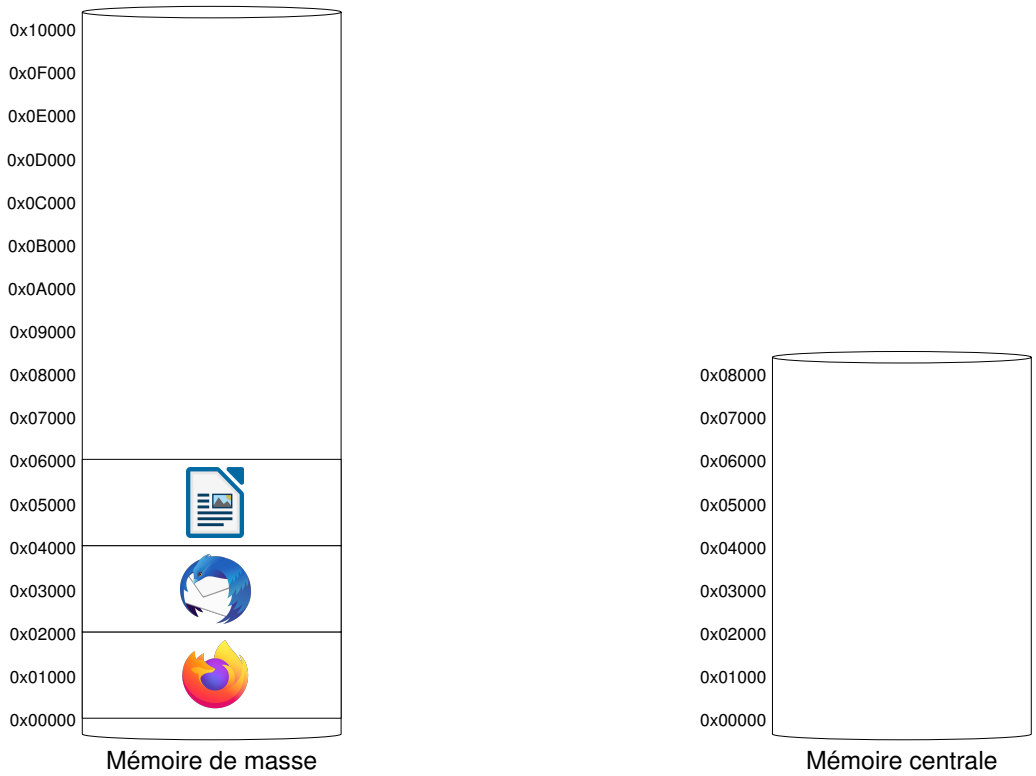
Gestion de la mémoire centrale



Comment le gérer en pratique ?

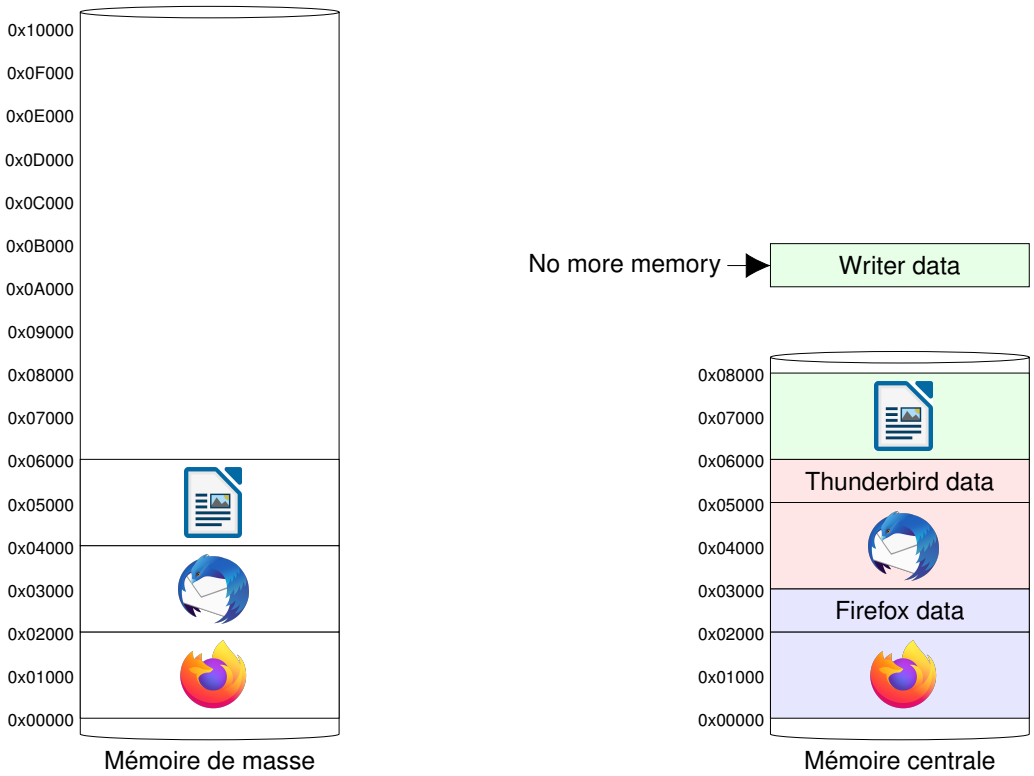


Proposition 1 : Eteindre le système ?

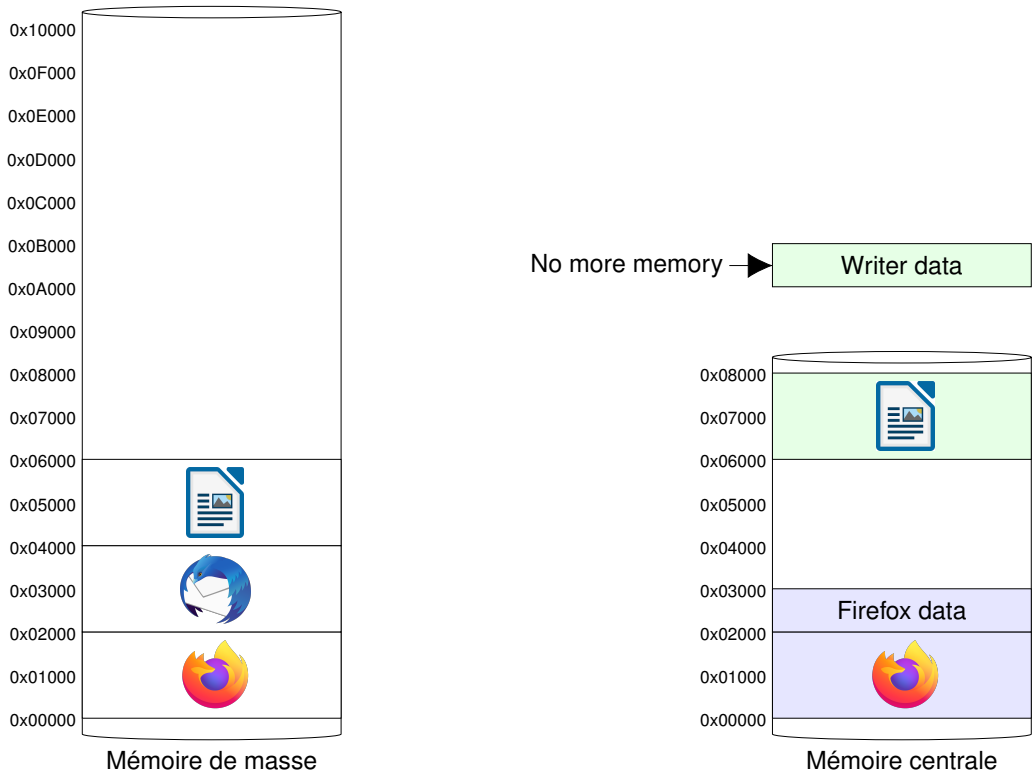


Proposition 1 : Eteindre le système ?

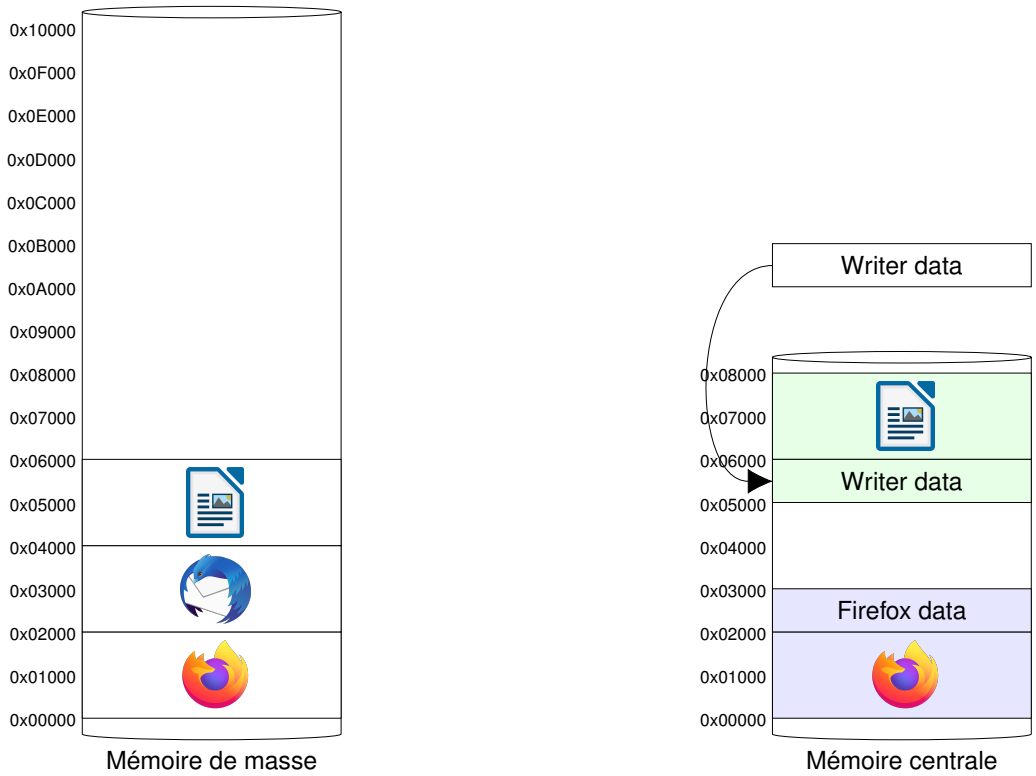
C'est le pire cas, système très instable avec perte de données et force le redémarrage de la machine, détruisant l'expérience utilisateur.



Proposition 2 : Fermer arbitrairement certaines applications ?

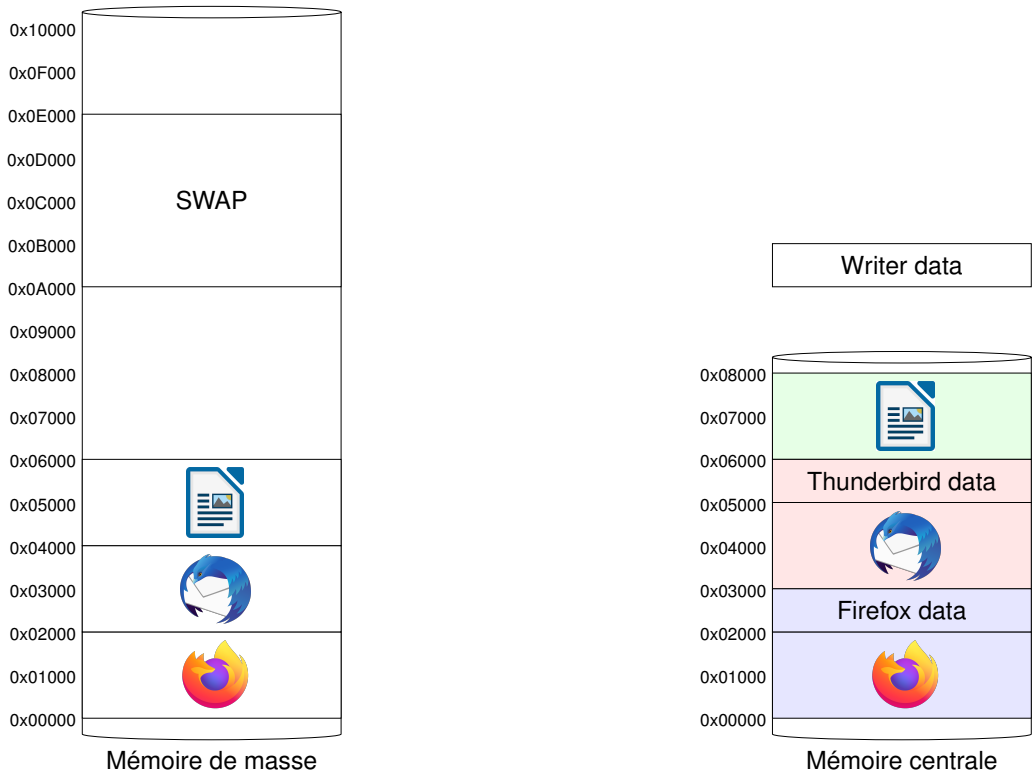


Proposition 2 : Fermer arbitrairement certaines applications ?

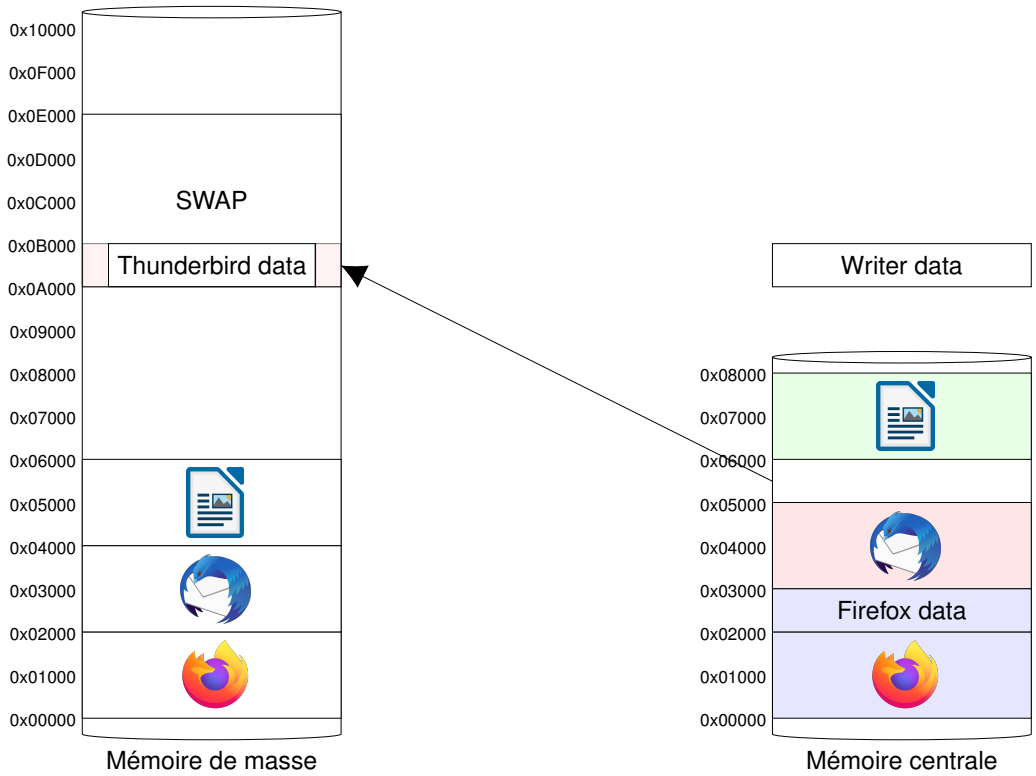


Proposition 2 : Fermer arbitrairement certaines applications ?

Risque de perte de données et risque d'une très mauvaise expérience utilisateur.

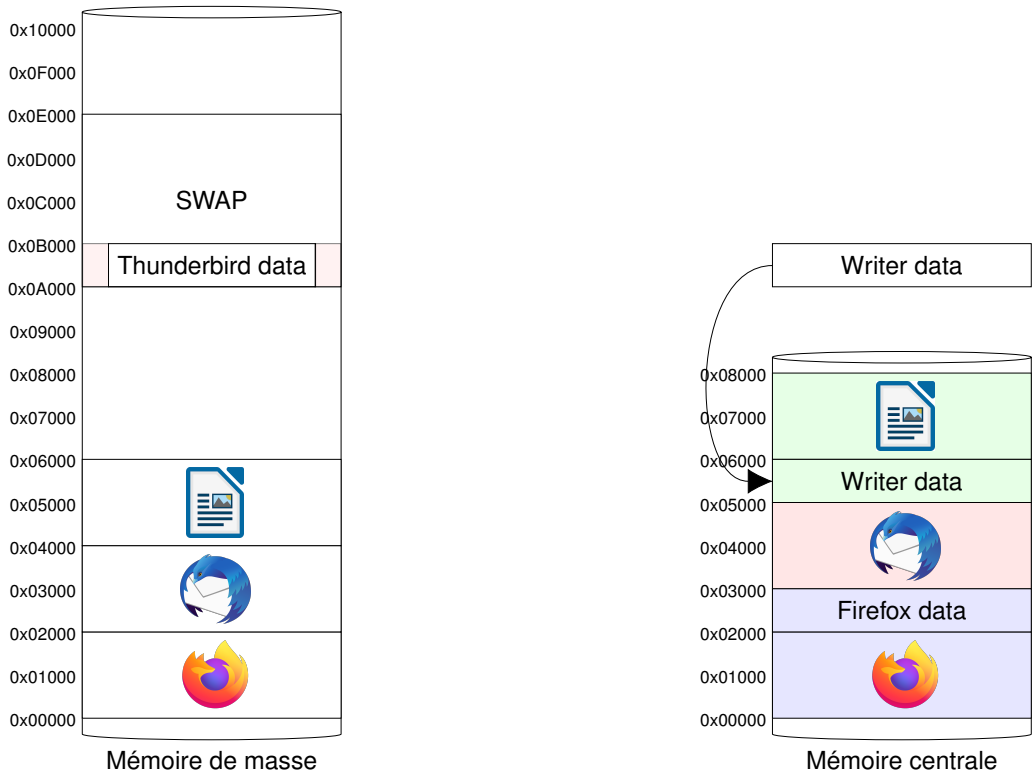


Proposition 3 : Copier temporairement des données dans la mémoire de masse ?



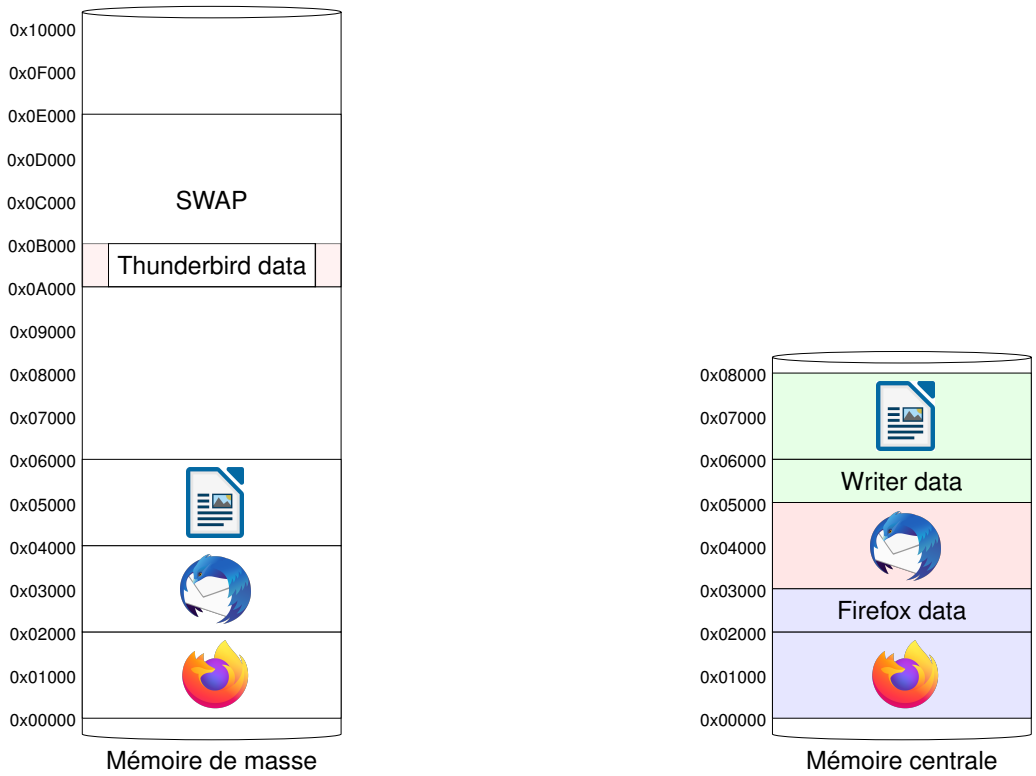
Proposition 3 : Copier temporairement des données dans la mémoire de masse ?

Oui, c'est le mécanisme du swap.



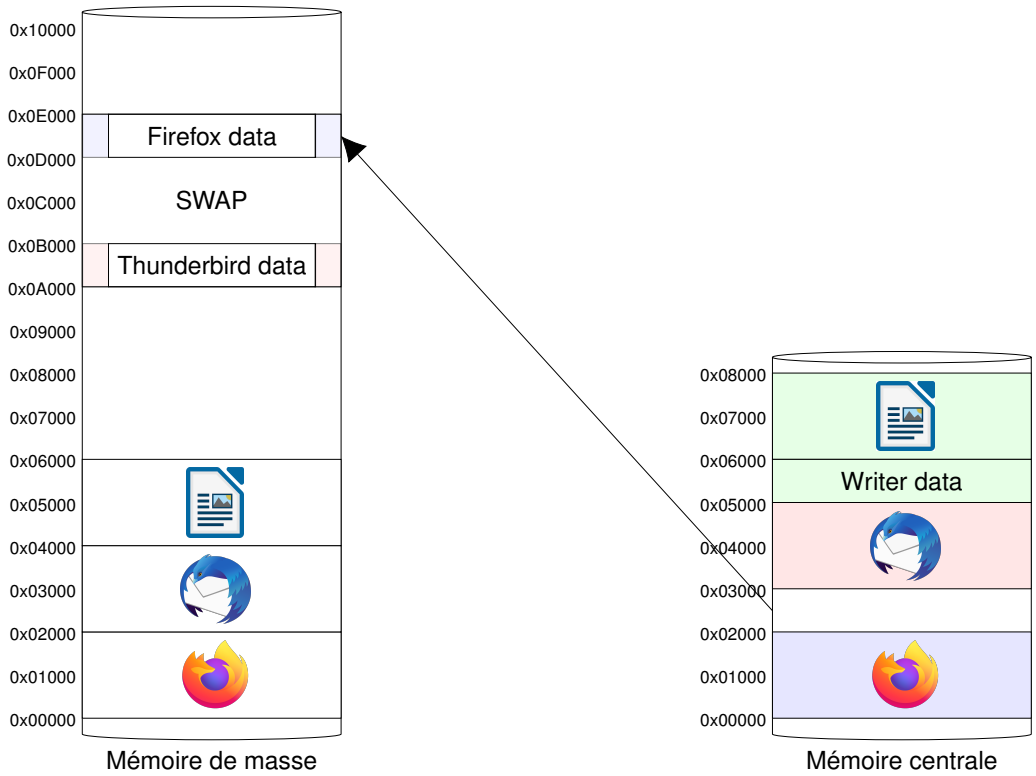
Proposition 3 : Copier temporairement des données dans la mémoire de masse ?

Oui, c'est le mécanisme du swap.



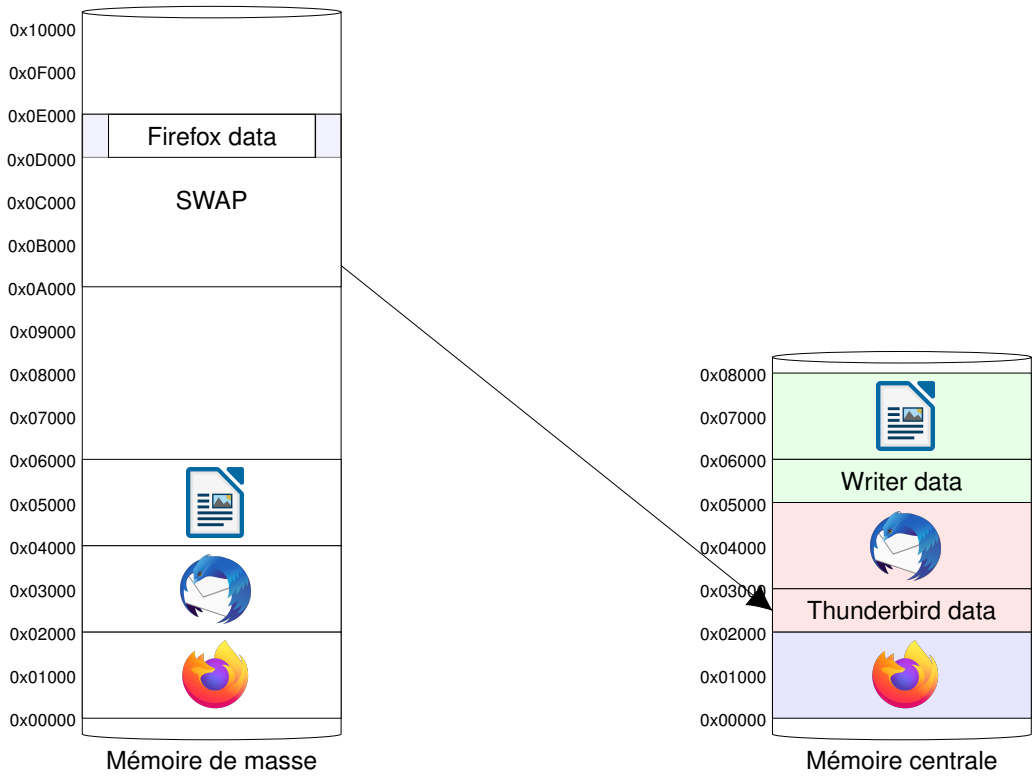
Mécanisme du SWAP

Et si vous retournez sur Thunderbird ?



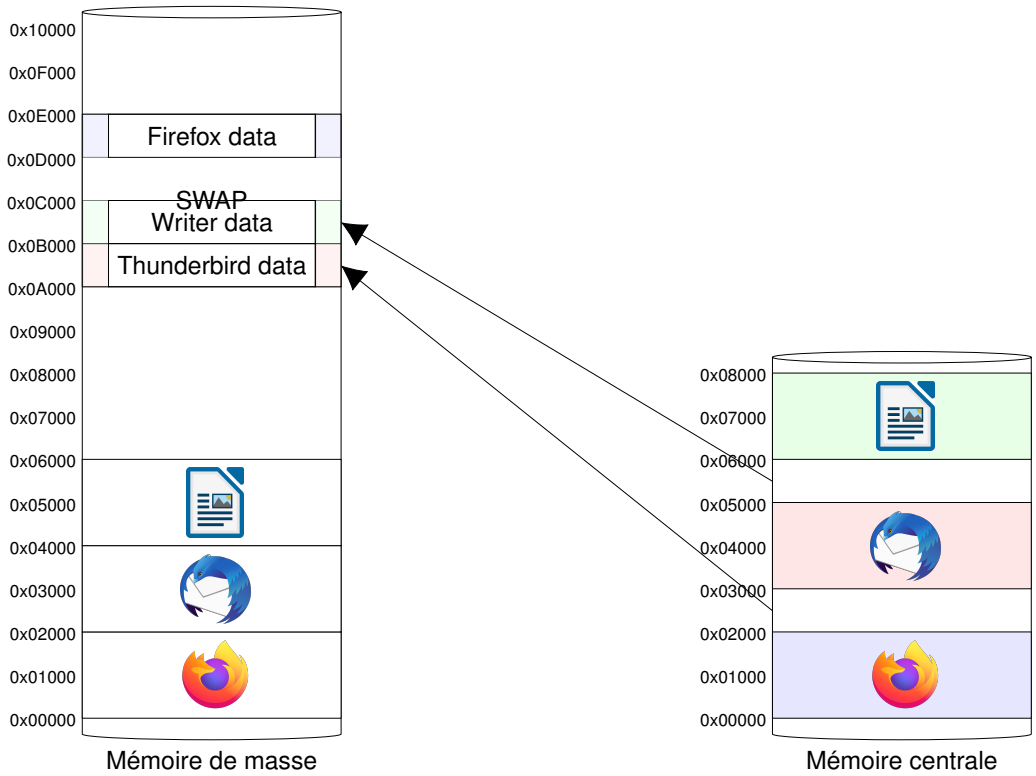
Mécanisme du SWAP

Et si vous retournez sur Thunderbird ?



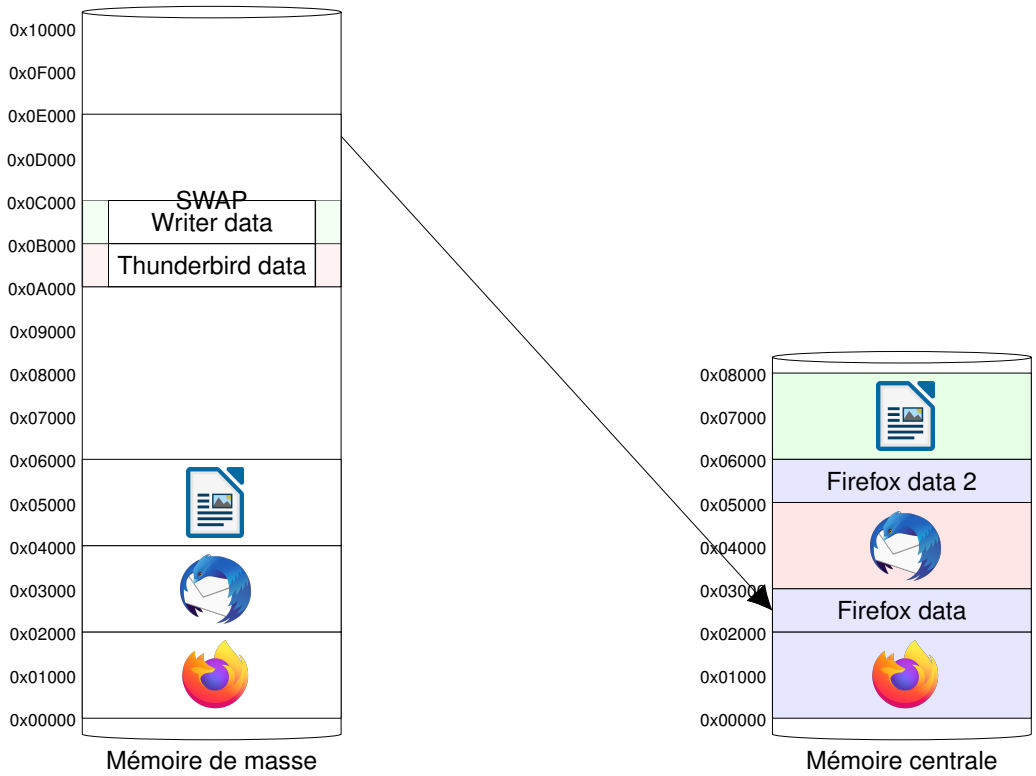
Mécanisme du SWAP

Et si vous retournez sur Thunderbird ?



Mécanisme du SWAP

Et si vous relancez Firefox et que vous téléchargez un gros fichier ?



Mécanisme du SWAP

Et si vous relancez Firefox et que vous téléchargez un gros fichier ?

- En principe, la mémoire SWAP = 2 x RAM.
- Le swapping se déclenche à partir d'un seuil, définit en % de RAM.

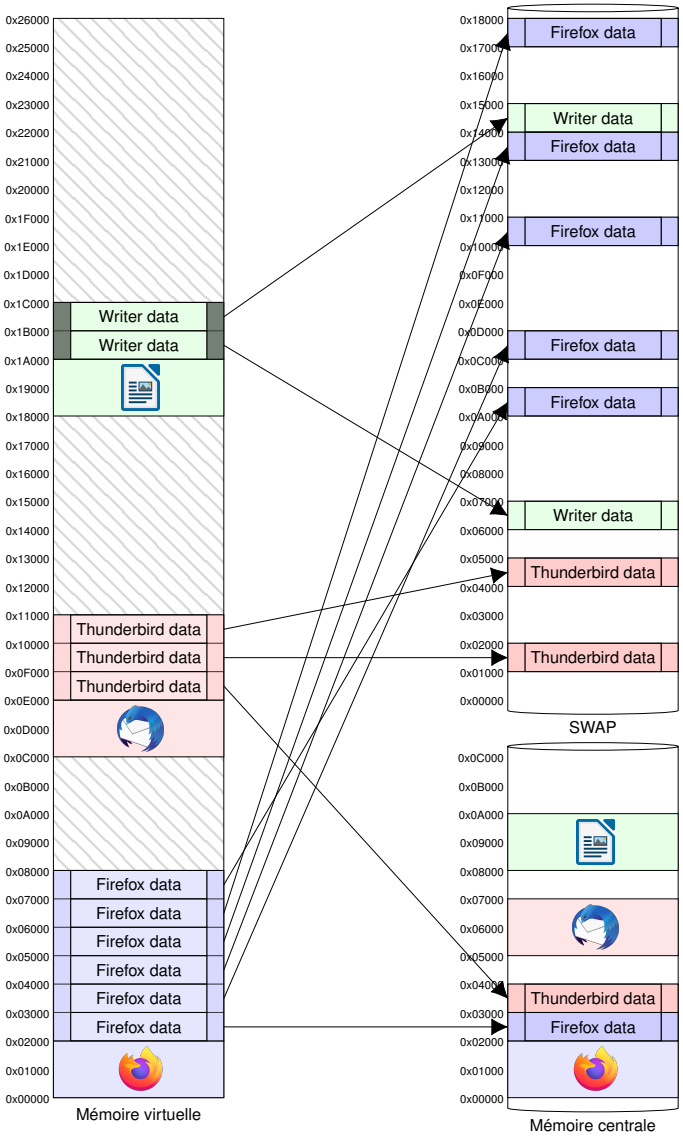
Commandes du terminal

- Visualisation du swapiness :
  cat /proc/sys/vm/swappiness
- Visualisation de la partition dédiée au SWAP :
  sudo gparted

Question

Que vaut en général le swappiness ?

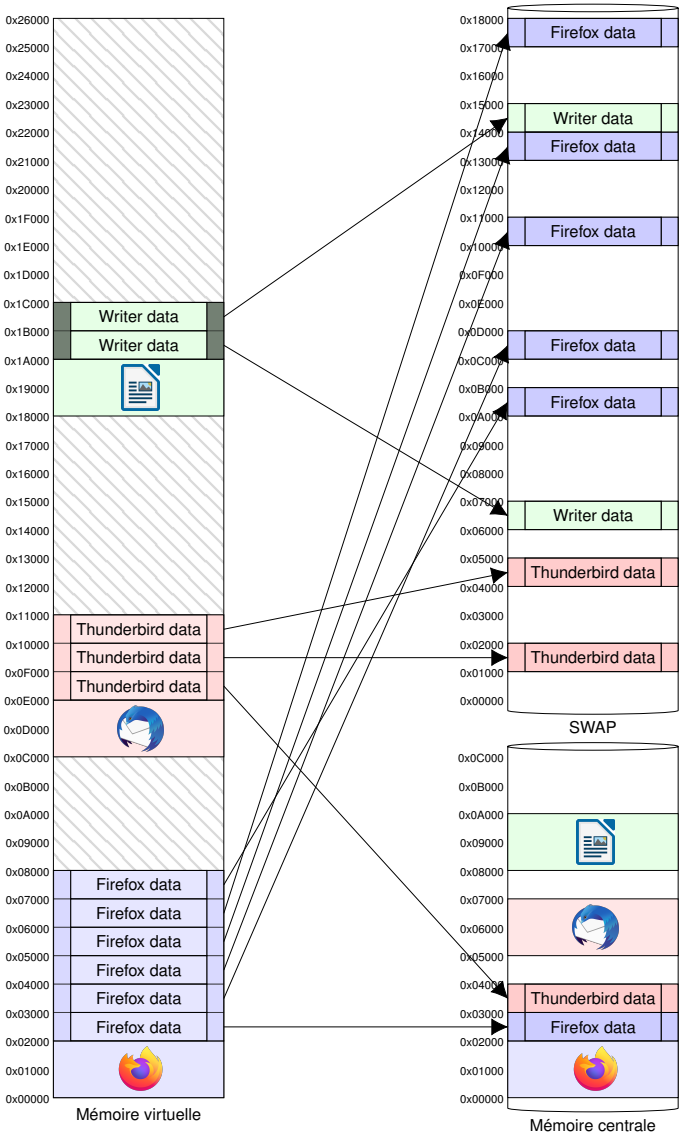
Comment le système d'exploitation garde une trace de la position précise des données ?



Mémoire virtuelle

Pour simplifier la gestion de la mémoire, le système d'exploitation gère une mémoire virtuelle qui contient toutes les données (mémoire centrale et SWAP).

Mémoire virtuelle : 1er essai



Propriétés

- C'est une mémoire contiguë, bien plus grande que la mémoire centrale.
- La mémoire virtuelle est découpée en pages de 4096 octets.
- La mémoire centrale est découpée en cadres de pages de 4096 octets. Chaque cadre permet donc d'accueillir 1 page virtuelle.

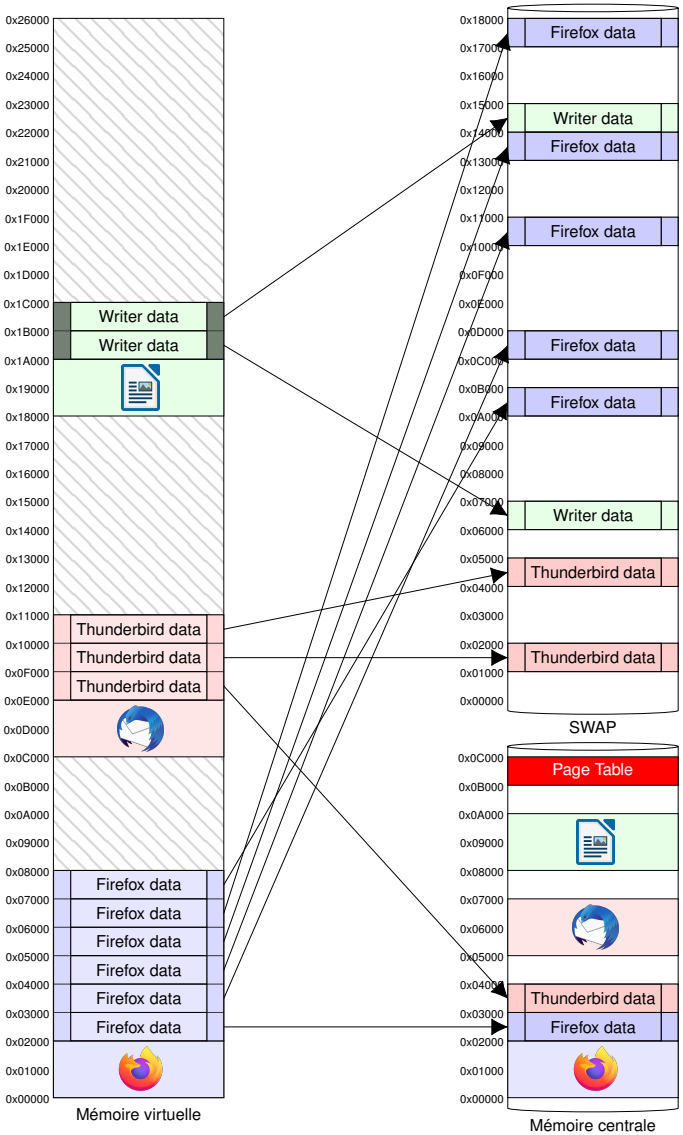
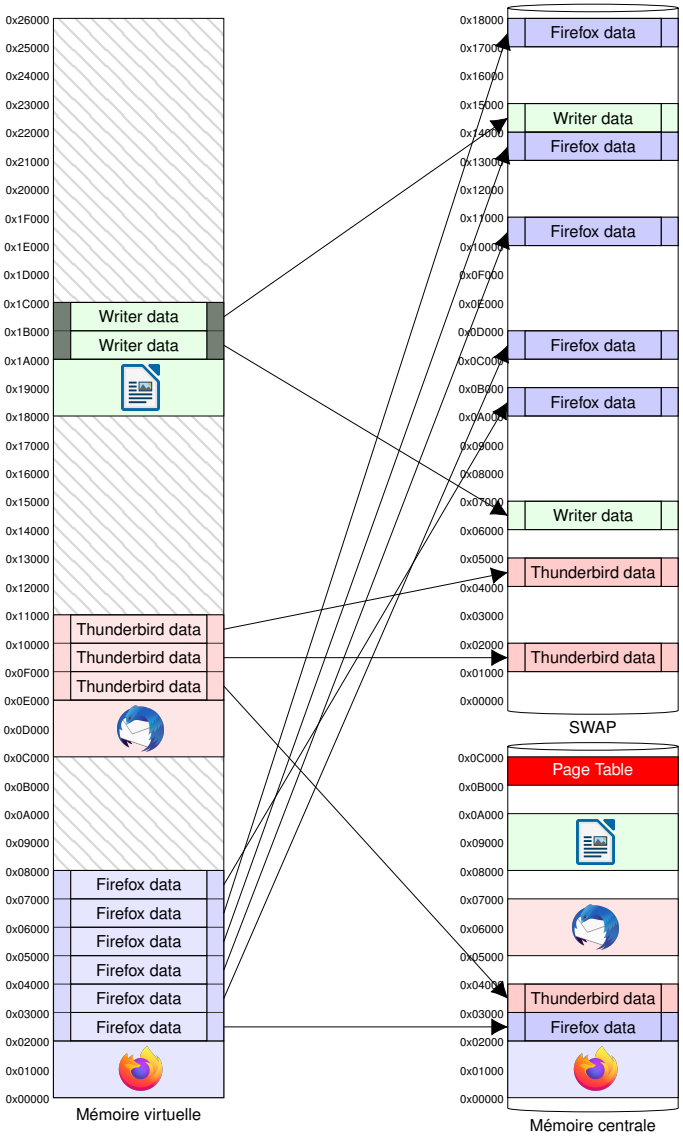


Table des pages

- L'ensemble des correspondances page/cadre est stocké dans un espace mémoire nommé table de page.
- Elle est stockée en mémoire centrale. (ici en rouge)

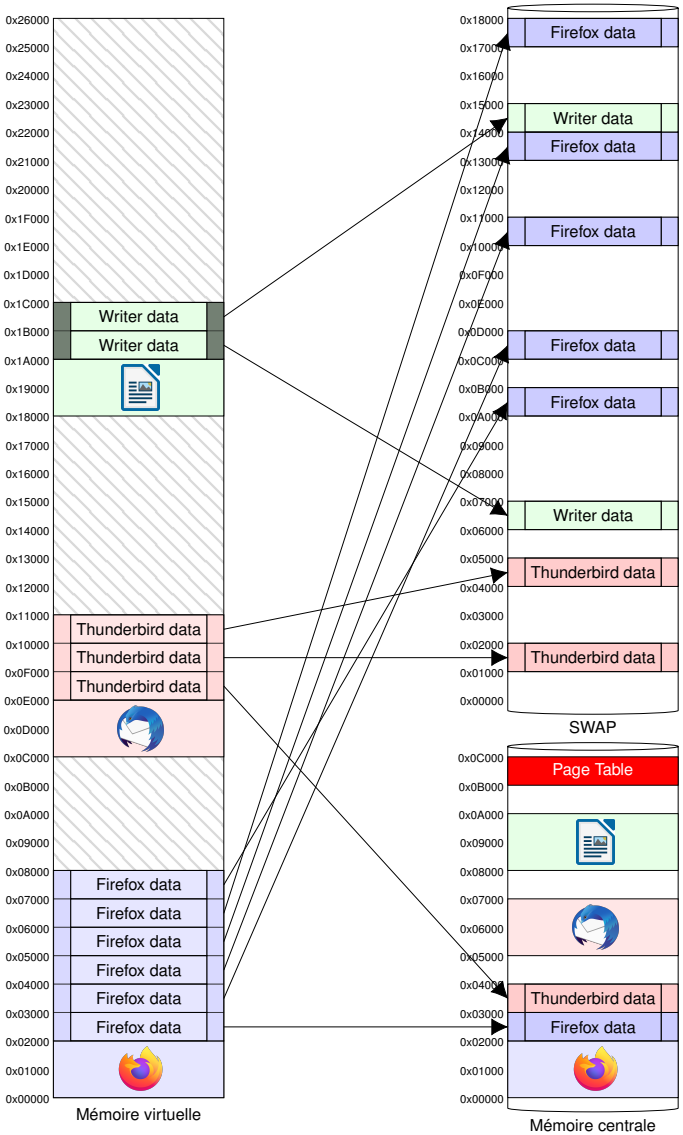
Mémoire virtuelle : 1er essai



Question

A votre avis, pourquoi avoir découpé la mémoire virtuelle et centrale en pages ?

Mémoire virtuelle : 1er essai



Question

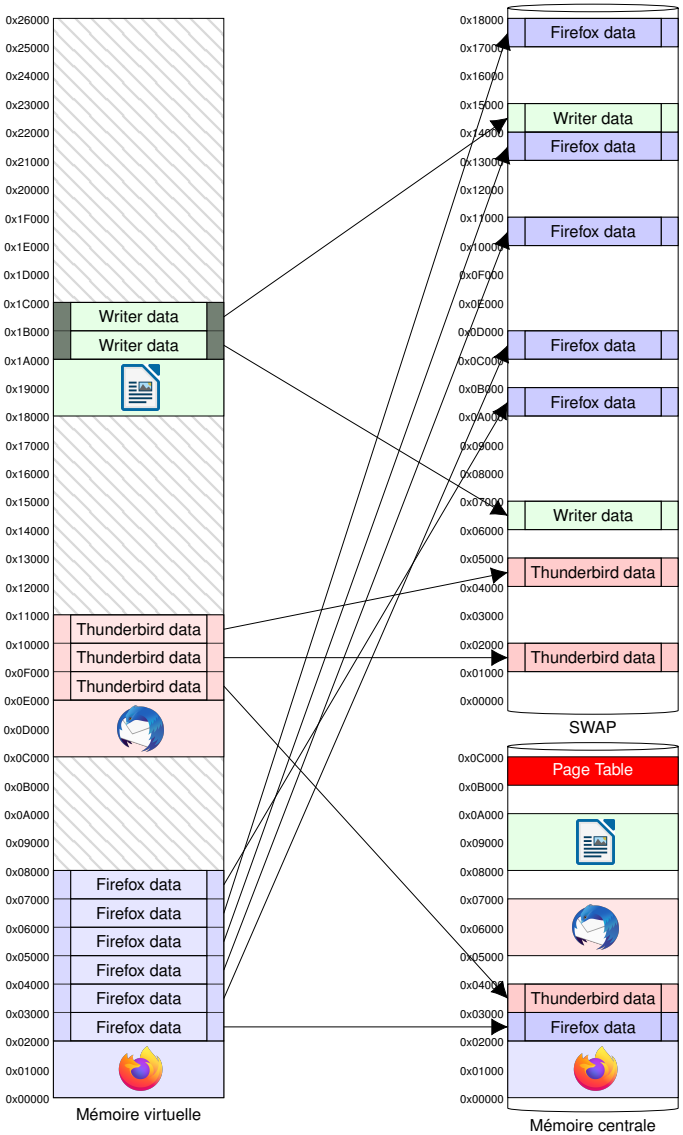
A votre avis, pourquoi avoir découpé la mémoire virtuelle et centrale en pages ?

Réponse

Pour simplifier la gestion de cette cartographie de la mémoire par le système d'exploitation.
Exemple pour 32 bits d'adresses virtuelles :

- Sans pagination : 4.2 milliards d'adresses
- Avec pagination : 16 millions de pages

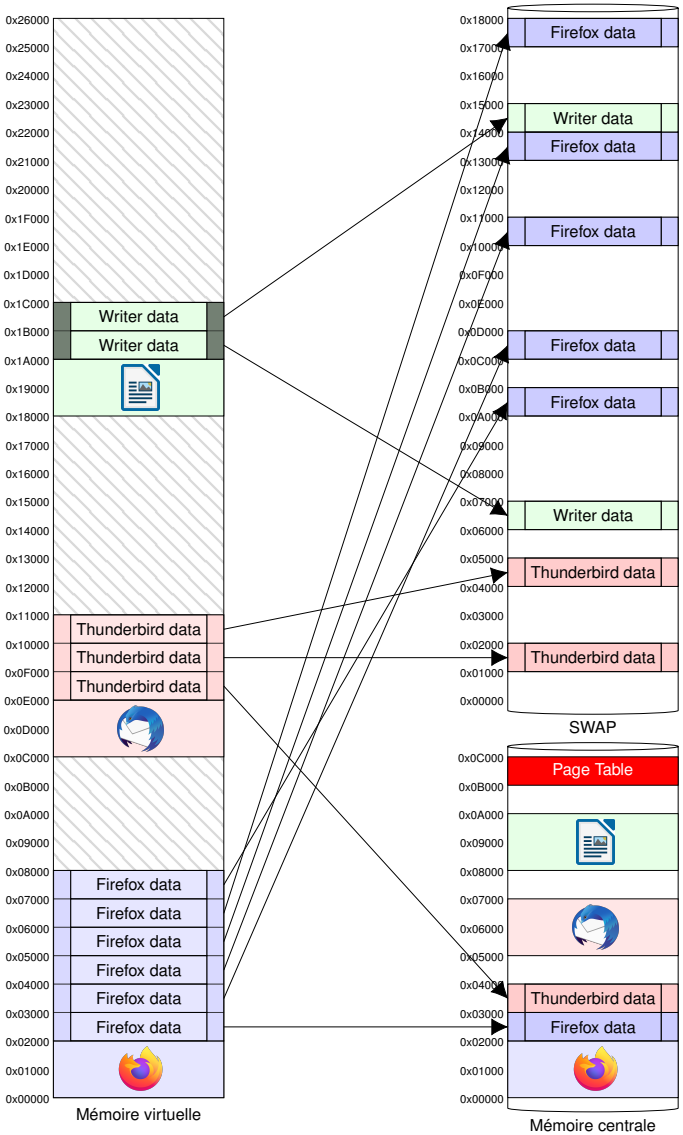
Mémoire virtuelle : 1er essai



Question

A votre avis, une application a-t-elle accès à la totalité de la mémoire virtuelle ?

Mémoire virtuelle : 1er essai



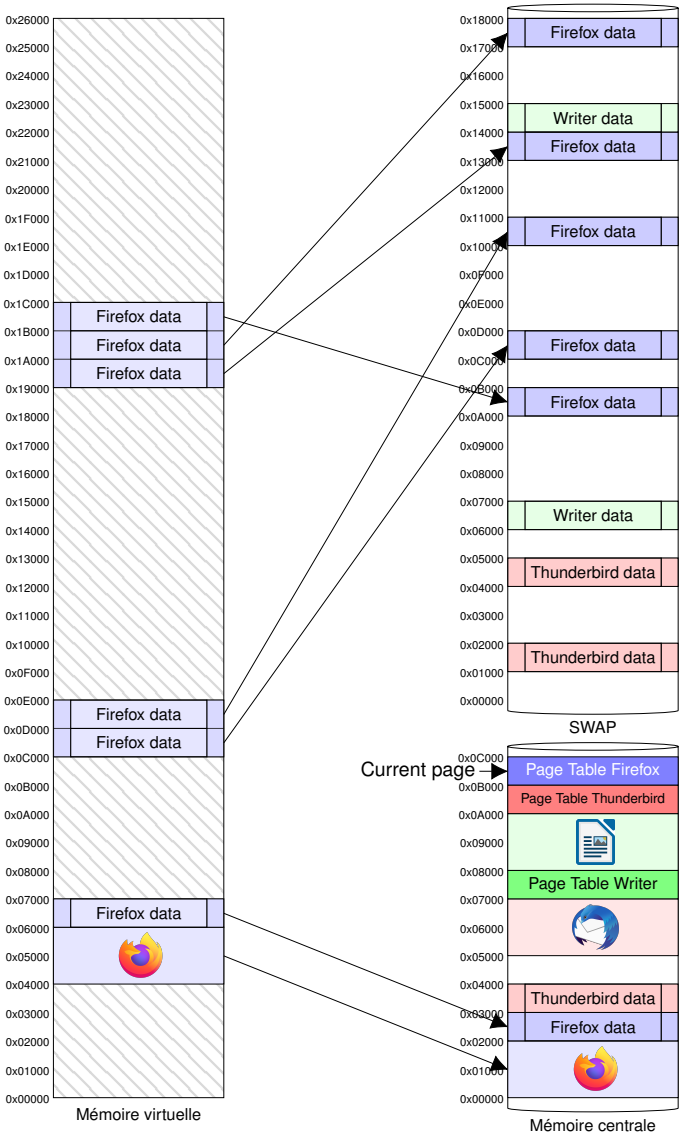
Question

A votre avis, une application a-t-elle accès à la totalité de la mémoire virtuelle ?

Réponse

Non, sinon elle pourrait modifier les autres applications qui tournent sur la machine !

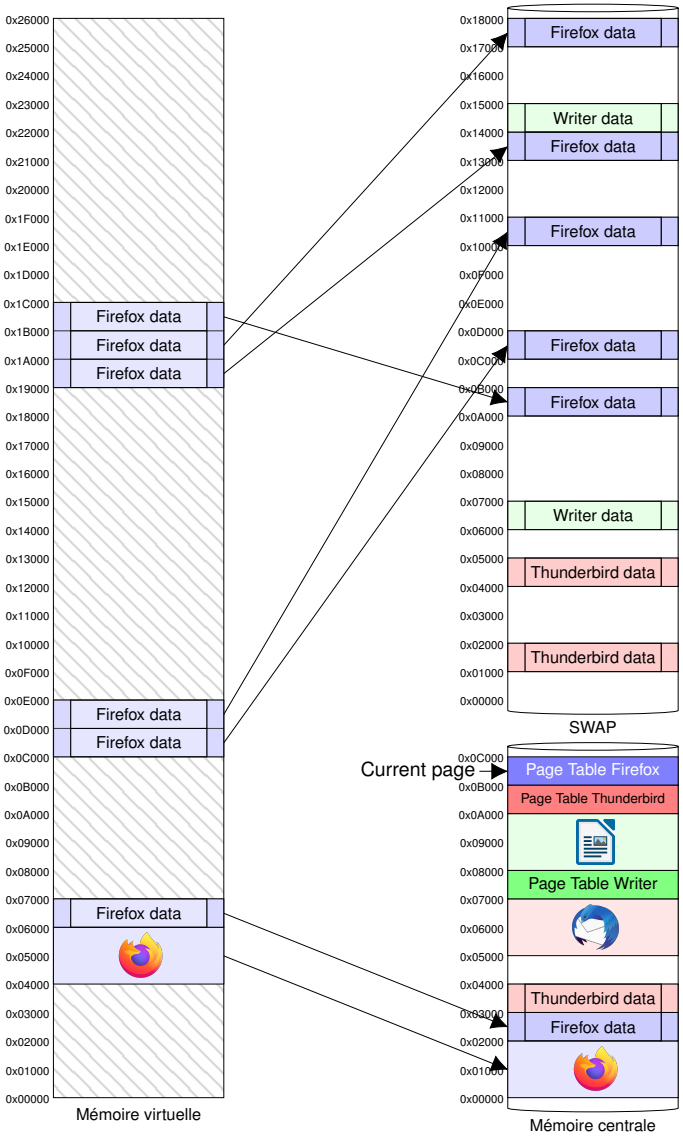
Avoir une seule mémoire virtuelle est-il une bonne solution ?



Apprendre par coeur

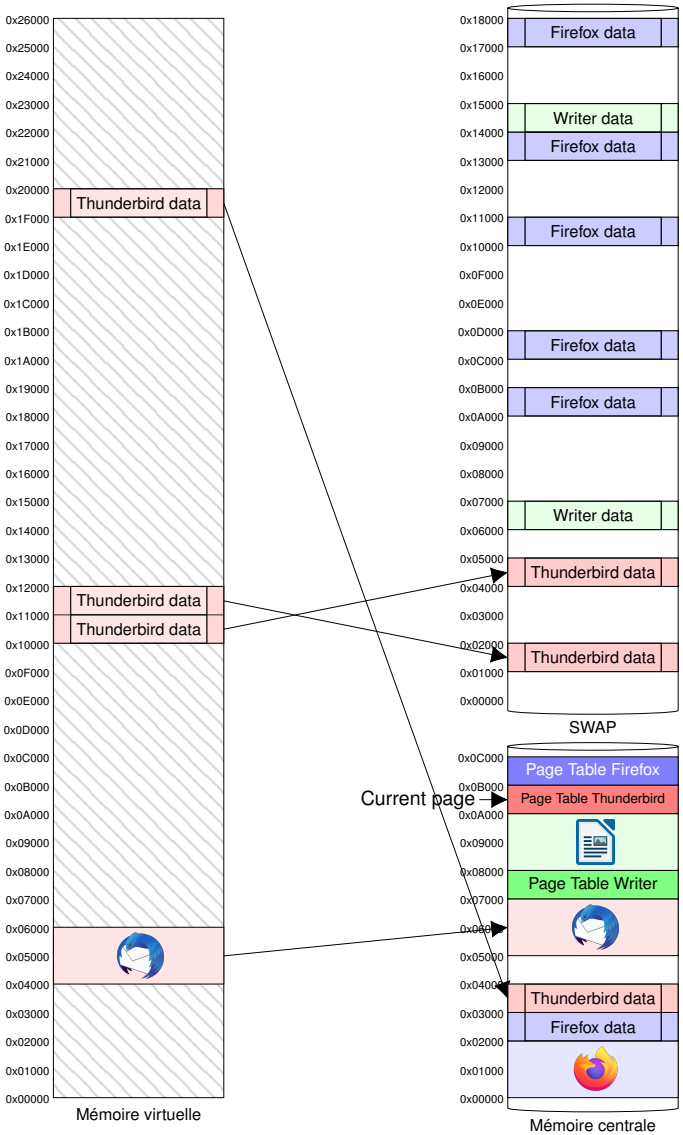
- Dans la pratique, on définit un espace d'adressage virtuel par application. Il y a donc une table des pages par application.
- Sauf exception rare, l'application dispose librement de son espace d'adressage virtuel.
- Pour garantir l'isolation entre les applications, les cadres de pages des autres applications ne sont pas associés à des pages de l'application actuelle.

Mémoire virtuelle : En pratique

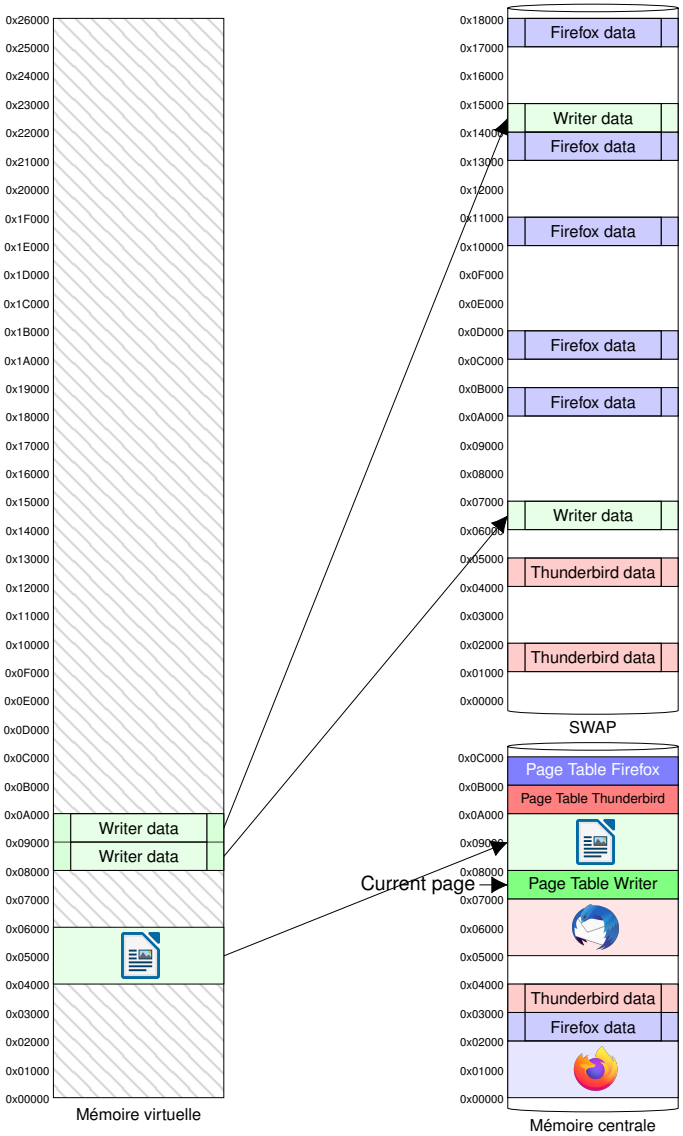


Correspondances lorsque Firefox s'exécute.

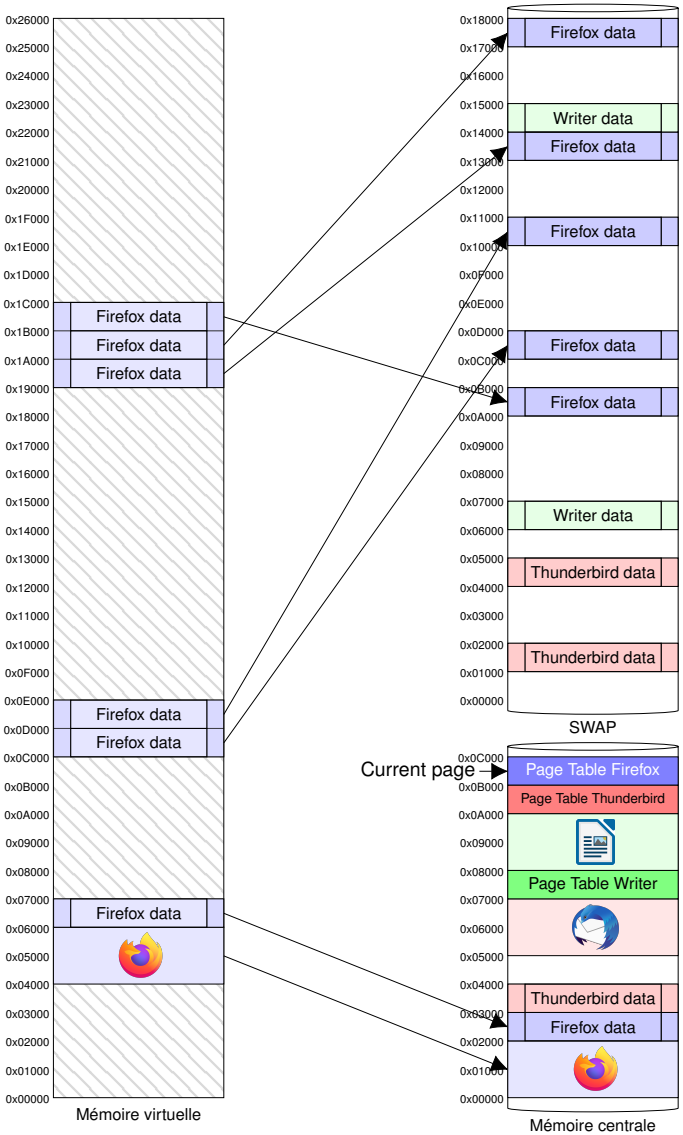
Mémoire virtuelle : En pratique



Correspondances lorsque Thunderbird s'exécute.

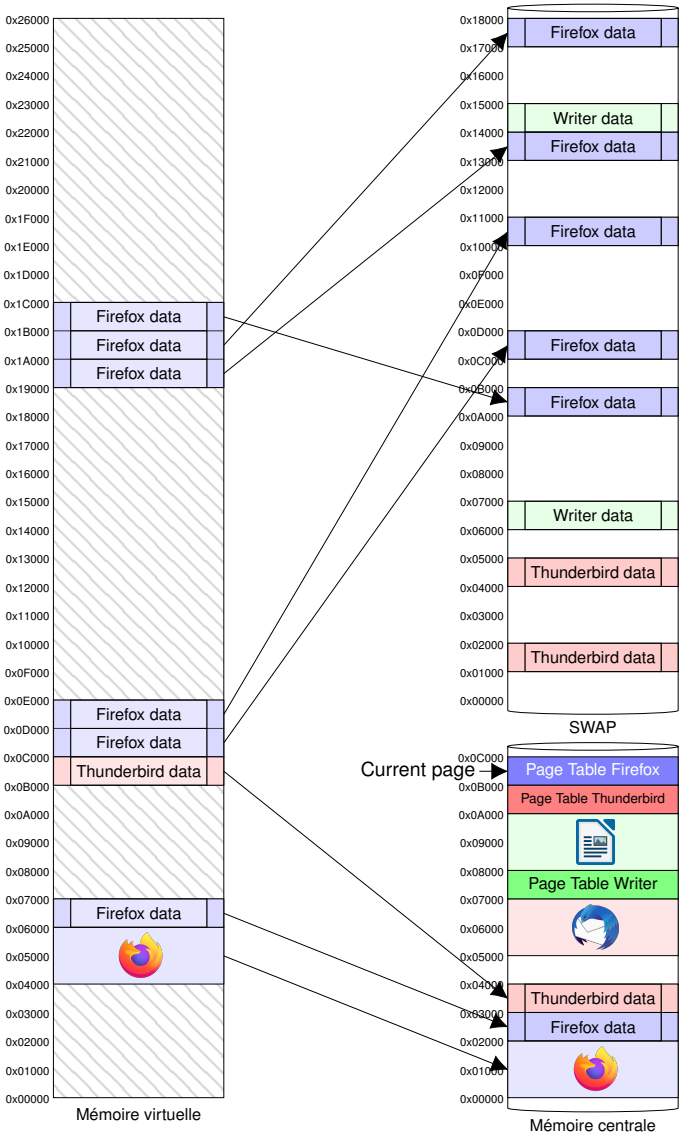


Correspondances lorsque Writer s'exécute.



Question

A votre avis, une application peut-elle modifier sa propre table des pages ?



Question

A votre avis, une application peut-elle modifier sa propre table des pages ?

Réponse

Surtout pas ! Sinon, elle pourrait librement associer les cadres de pages des autres applications à ses propres pages virtuelles !

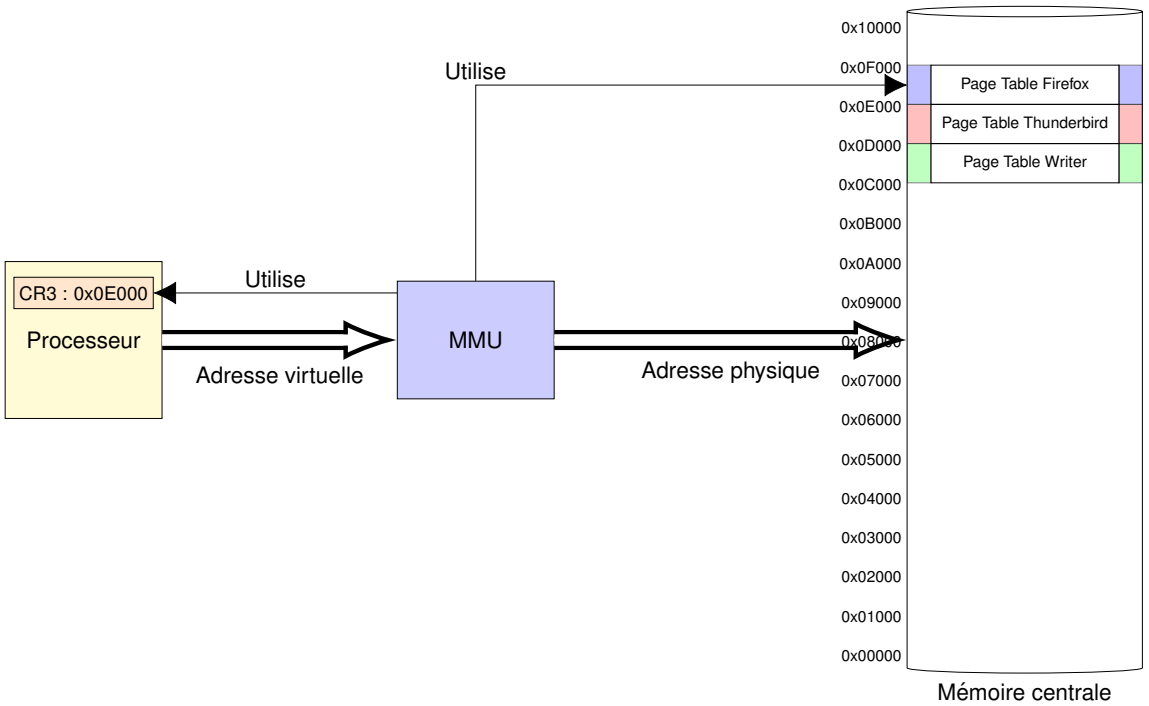
Pour la suite, une application en train de s'exécuter sur la machine sera nommée processus

L'espace d'adressage virtuel d'un processus peut être visualisé depuis le terminal.

Commandes du terminal

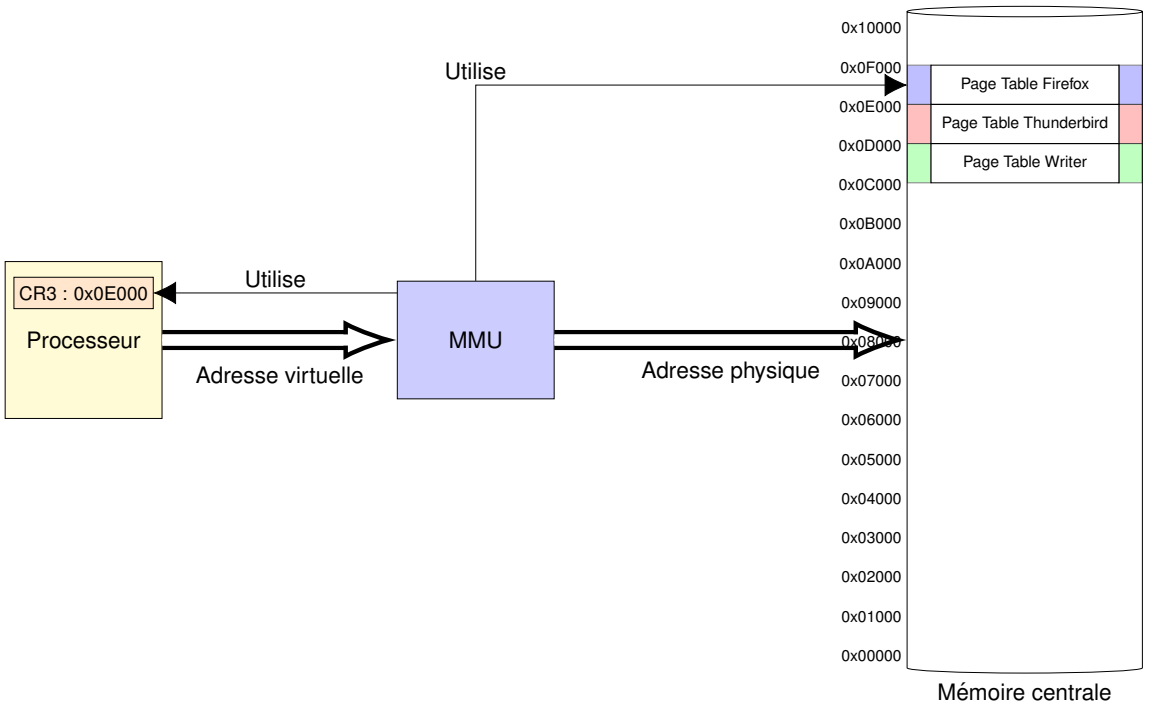
- Lancement de gedit en arrière plan :
  gedit &
- Récupération du numéro de processus de gedit :
  ps ax | grep gedit
- Visualisation de l'espace d'adressage virtuel de gedit :
  pmap *numero_de_processus_de_gedit*

Comment est gérée la traduction adresse virtuelle/adresse physique matériellement ?



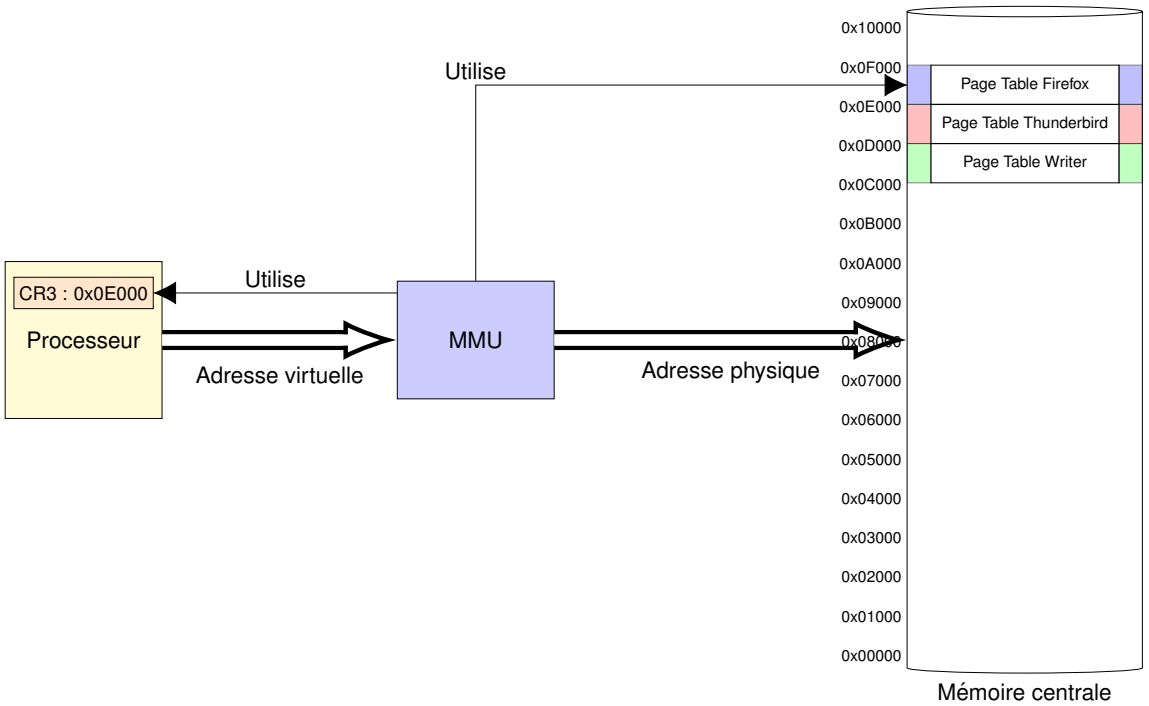
La *Memory Management Unit* (MMU)

La MMU est l'unité matérielle permettant la traduction d'une adresse virtuelle vers l'adresse physique.



Mécanisme de traduction [1]

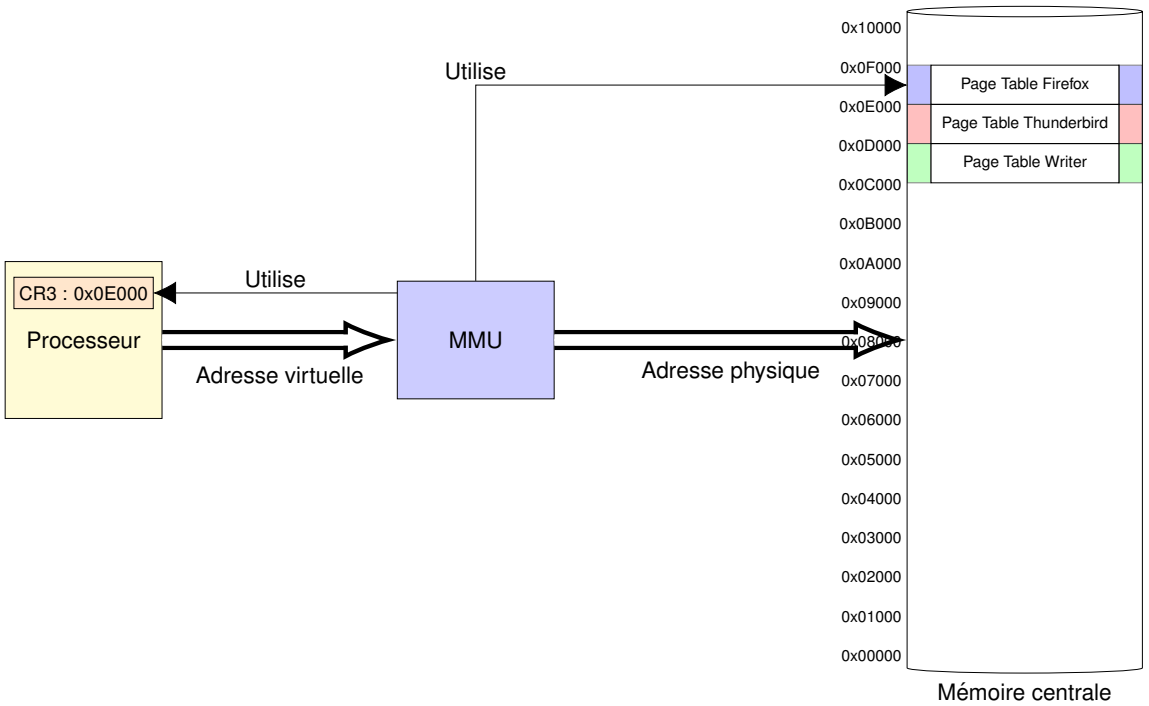
Pour traduire une adresse virtuelle en adresse physique, la MMU a besoin de connaitre l'adresse en mémoire centrale de la table des pages du processus en train de s'exécuter.



Mécanisme de traduction [2]

Un registre dédié du processeur, le registre CR3, stocke cette adresse (qui est une adresse physique).

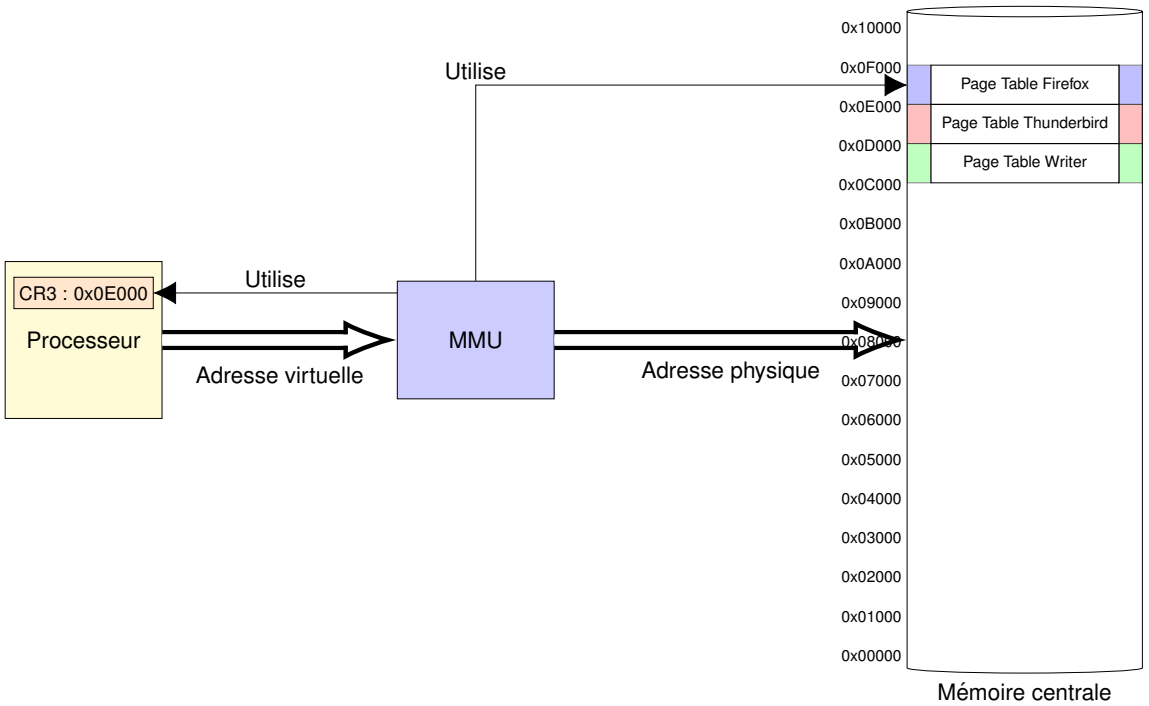
Mécanisme matériel associé



Question

Un processus peut-il modifier la valeur du registre CR3 ?

Mécanisme matériel associé

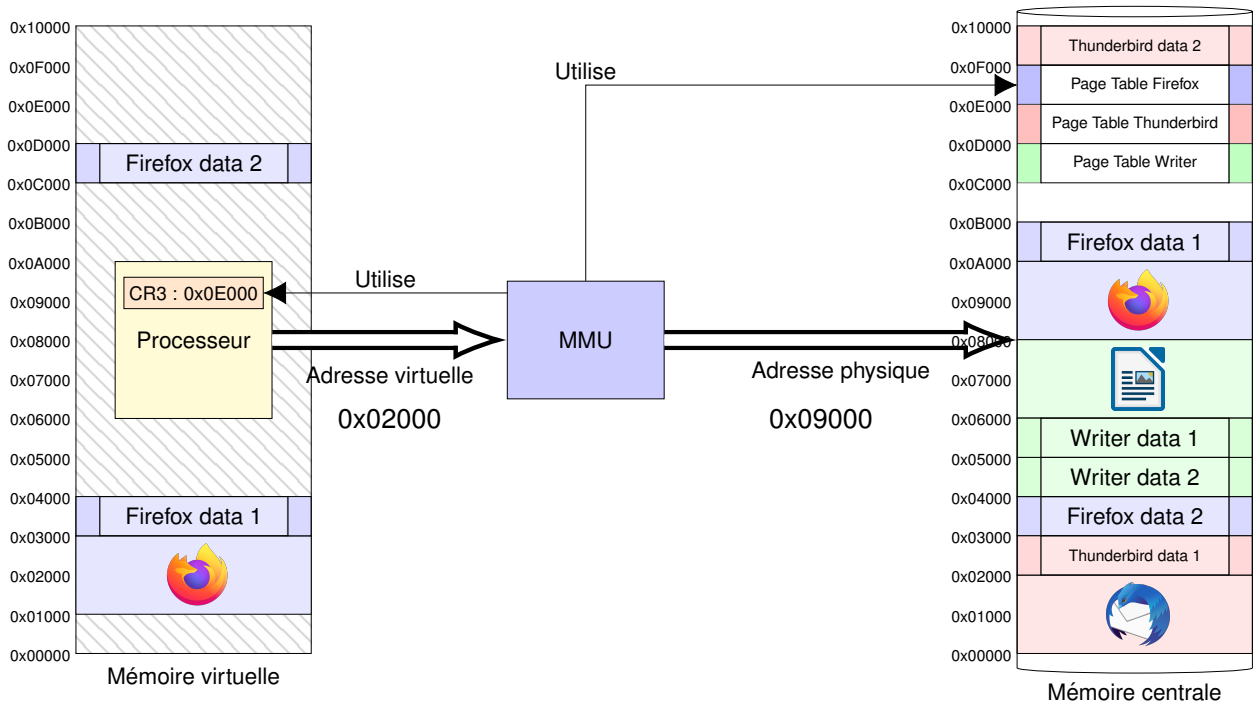


Question

Un processus peut-il modifier la valeur du registre CR3 ?

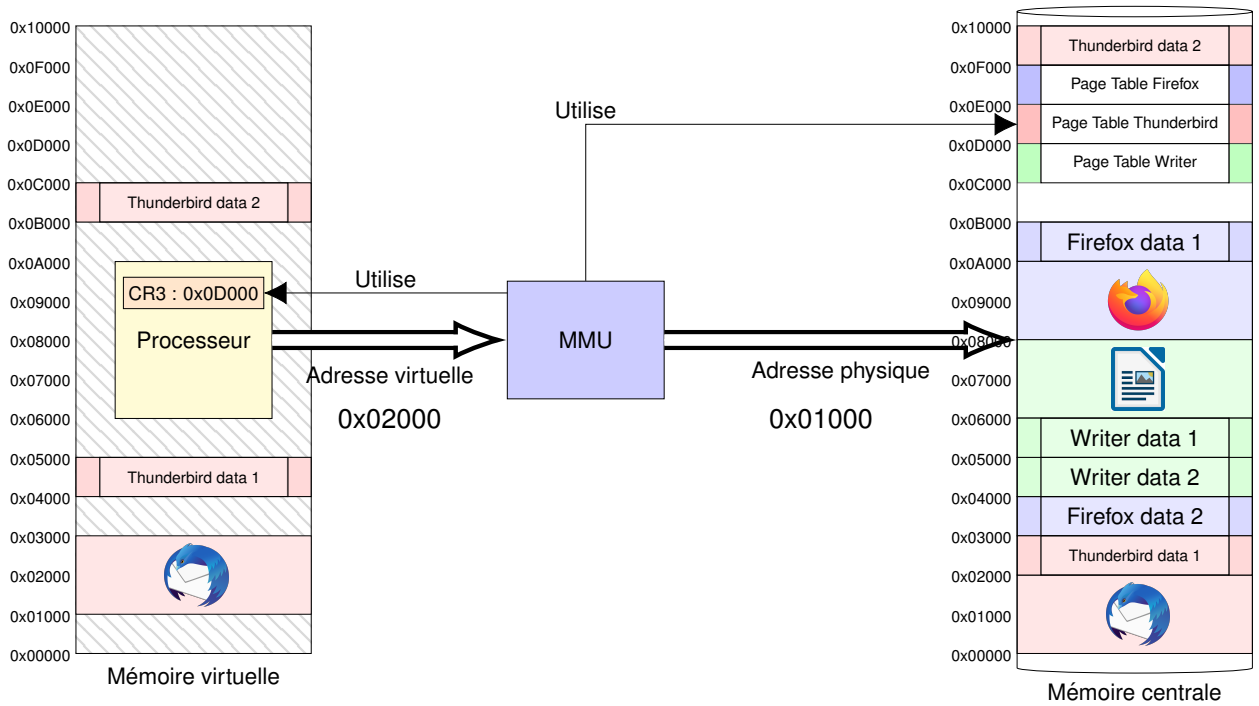
Réponse

Non, sinon il pourrait accéder aux données d'un autre processus!



Exemple 1: Firefox s'exécute

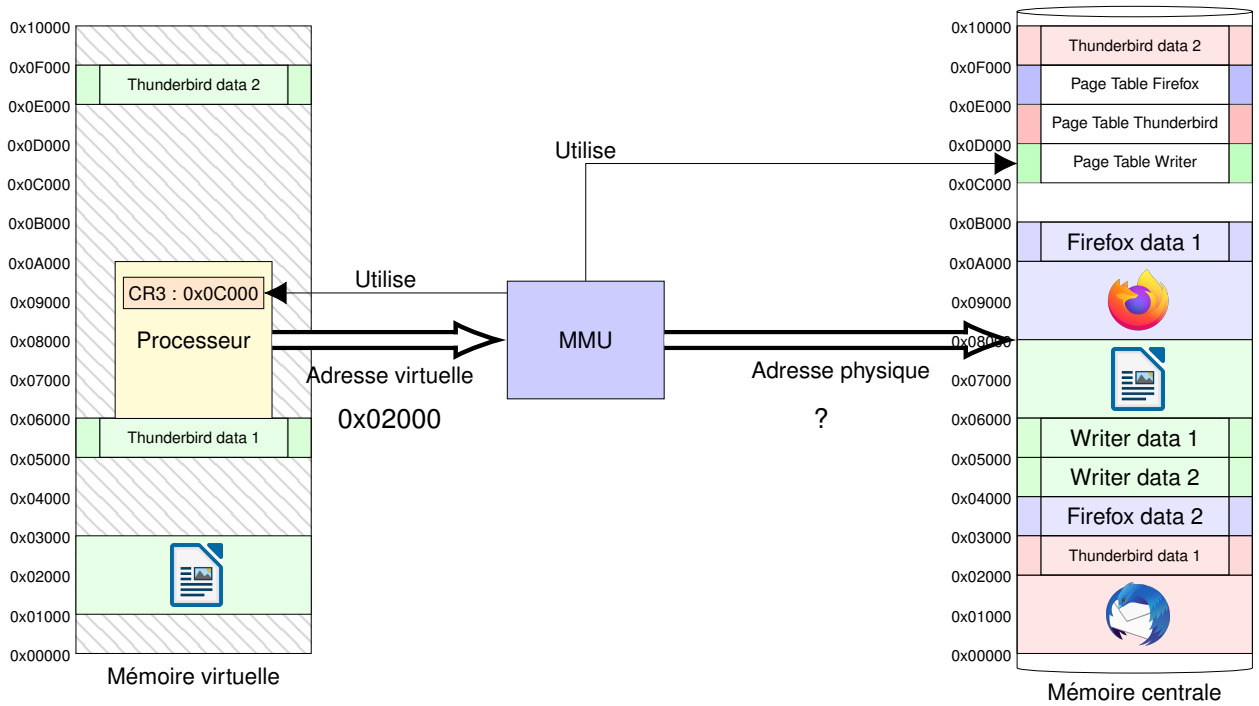
Firefox s'exécute, CR3 est positionné sur 0x0E000. Si Firefox demande l'adresse 0x02000 (dans son programme), alors la MMU, à l'aide de la table des pages à l'adresse 0x0E000, va traduire cette adresse en 0x09000.



Exemple 2: Thunderbird s'exécute

Maintenant Thunderbird s'exécute, CR3 a été positionné à la valeur 0x0D000, la MMU utilisera donc la table à l'adresse 0x0D000. Ainsi, pour la même adresse virtuelle 0x02000, elle est traduite maintenant en 0x01000.

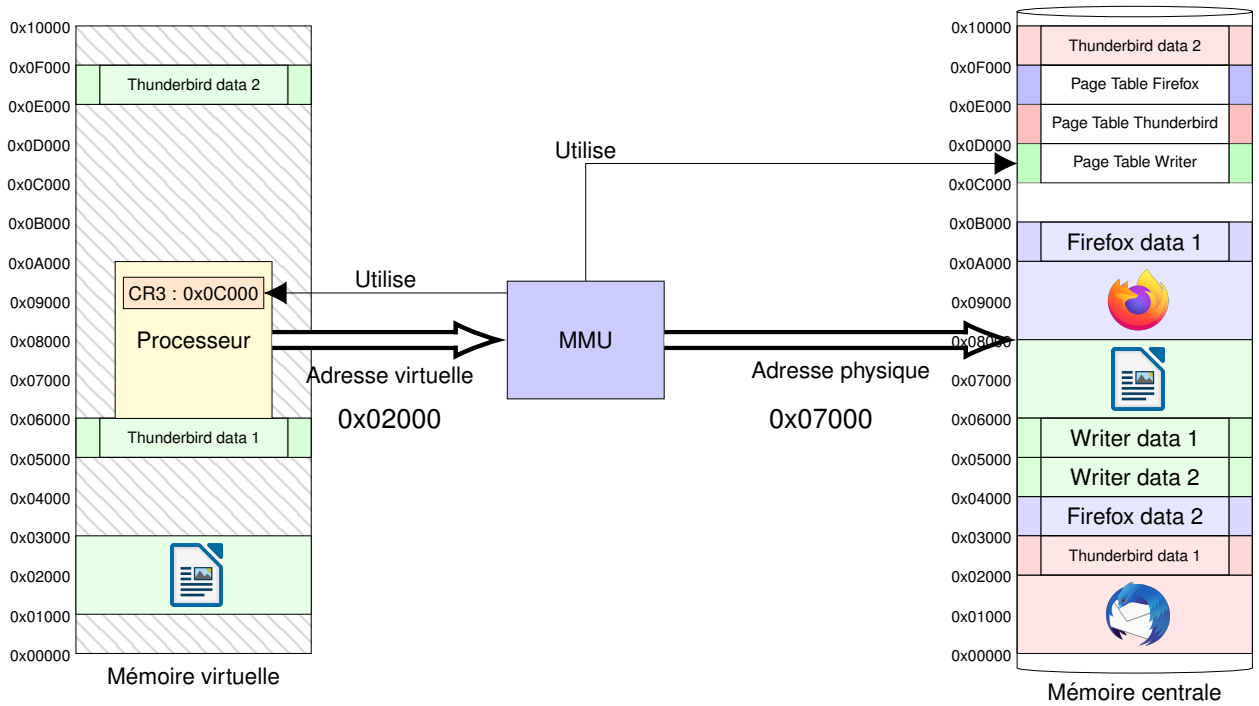
Mécanisme matériel associé



Question

Writer s'exécute, comment sera traduit 0x02000 ?

Mécanisme matériel associé



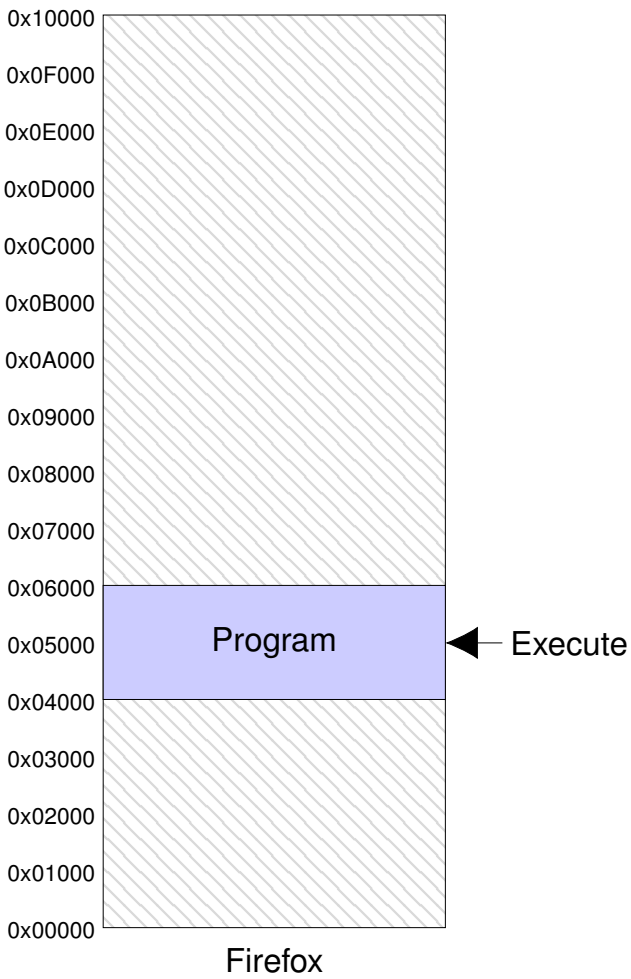
Question

Writer s'exécute, comment sera traduit 0x02000 ?

Réponse

Par 0x07000 !

Cette construction est-elle parfaitement sécurisée ?

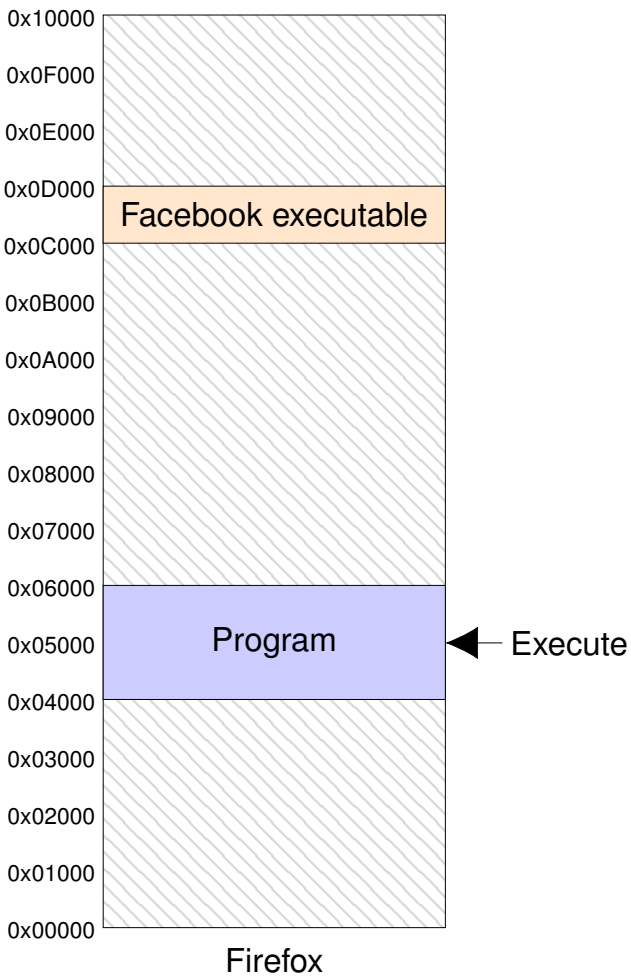


Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 1

Lancement du téléchargement de l'exécutable de Facebook.

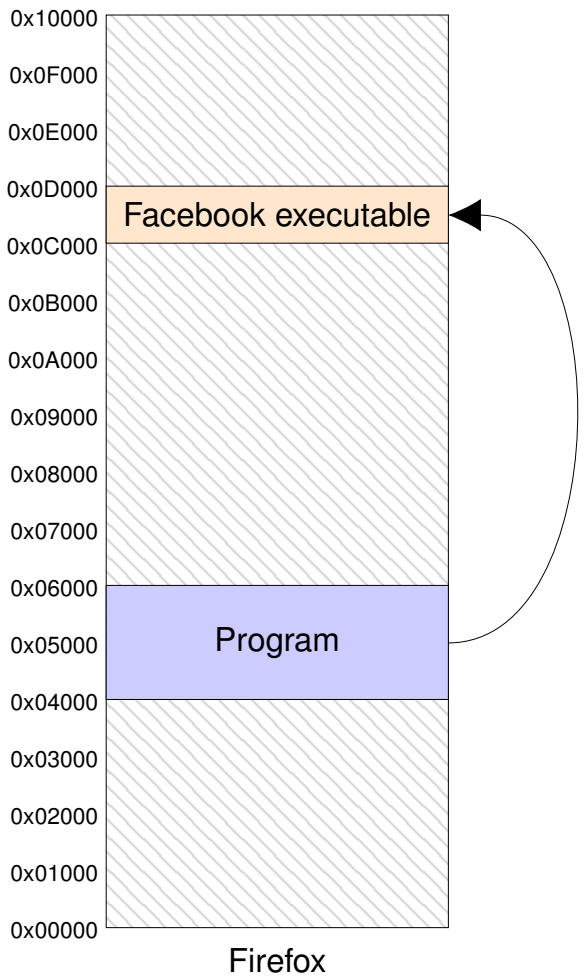


Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 2

Copie de l'exécutable dans une page libre.

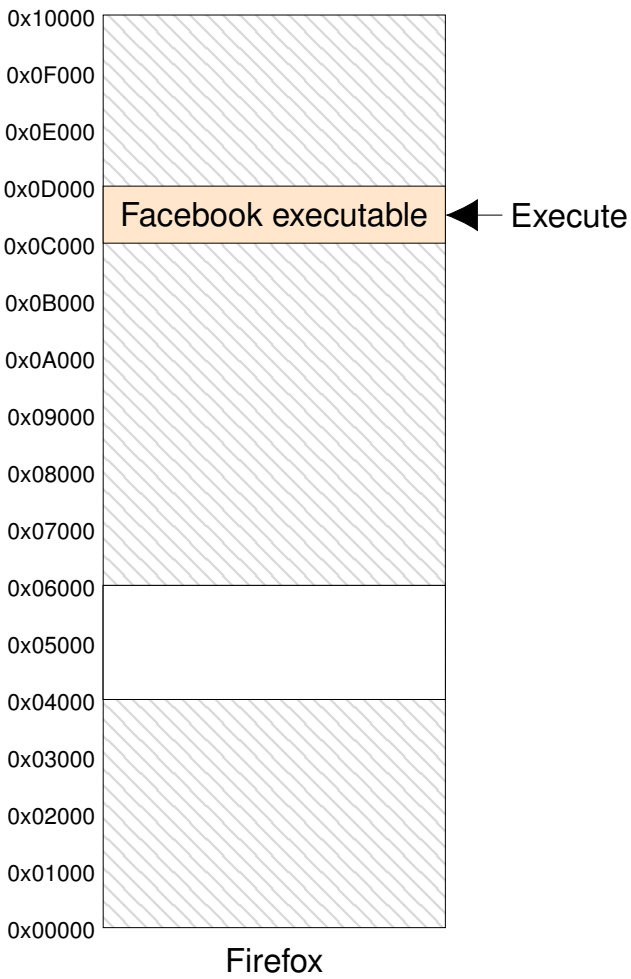


Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 3

Saut dans l'exécutable de Facebook.

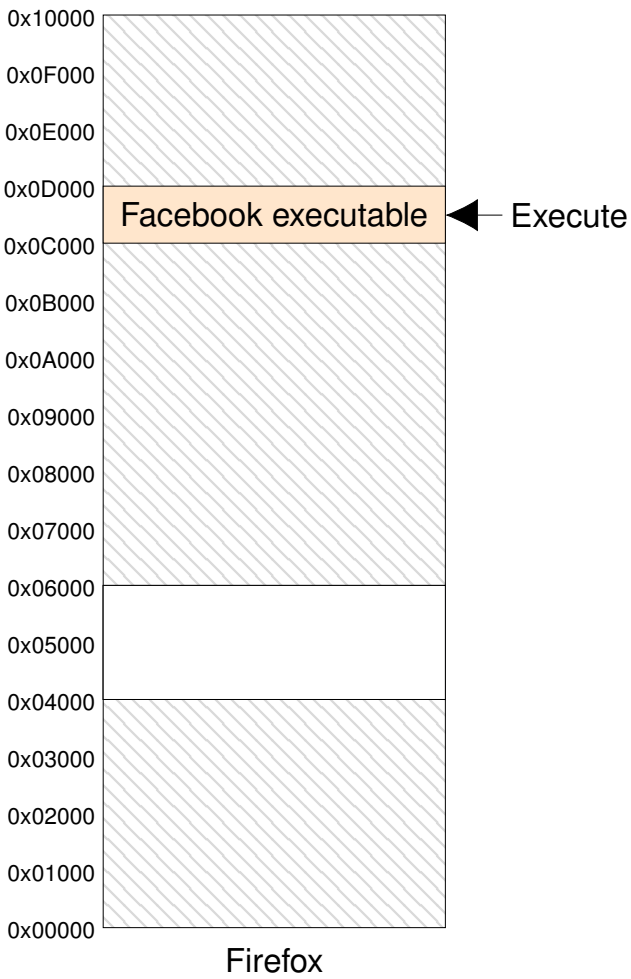


Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 4

Troll de Firefox en écrasant son programme.

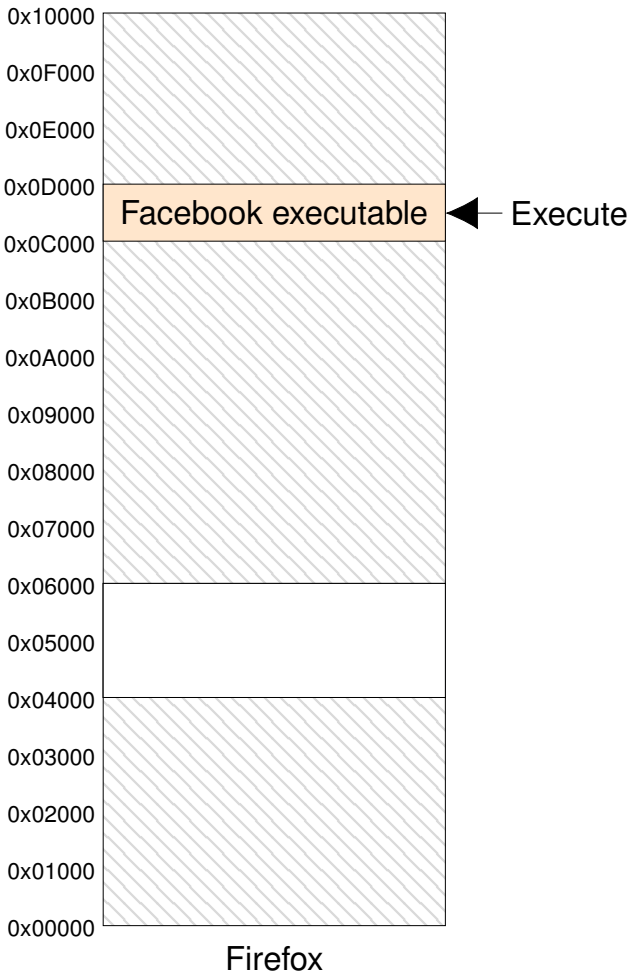


Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 5

Génération de trous noirs (c'est une blague).



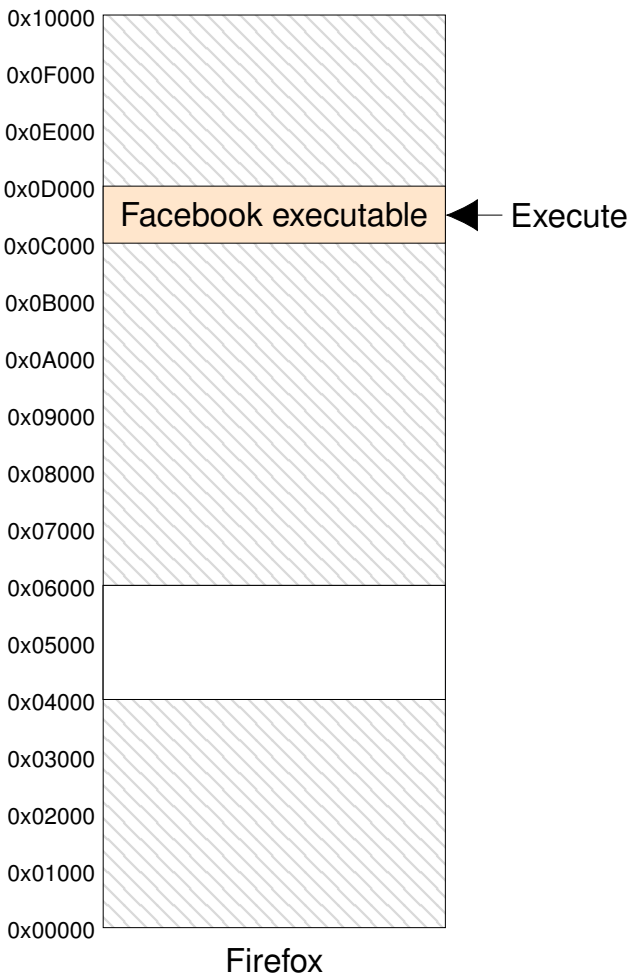
Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 6

Suppression des fichiers de la mémoire de masse ne demandant pas de droits administrateurs.

Isolation : Système sécurisé ?



Cas d'application

Vous naviguez sur le site de Facebook depuis votre navigateur. Que peut-il se passer sur votre ordinateur ?

Etape 7

Et si Firefox est exécuté avec les droits Administrateurs : Suppression de tous les fichiers de la mémoire de masse.

Quel est le coût réel de la virtualisation de la mémoire ?

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage

CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

Étudions le mécanisme de traduction sur 1 exemple. En principe, l'adresse virtuelle est découpée en 3 champs :

- Dossier : c'est une table contenant l'adresse (physique) d'un ensemble de tables de pages.
- Table : c'est une table contenant l'adresse (physique) d'un ensemble de pages.
- Décalage : c'est l'octet demandé dans la page.

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction

0x112

CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

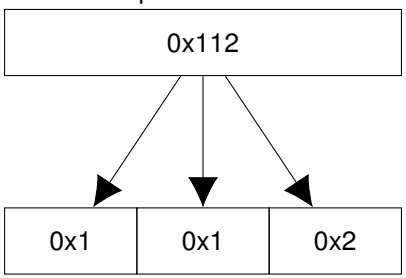
Pour fixer les idées, on considère que chaque champ est sur 4 bits. On souhaite traduire l'adresse virtuelle 0x112.

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

Pour fixer les idées, on considère que chaque champ est sur 4 bits. On souhaite traduire l'adresse virtuelle 0x112.

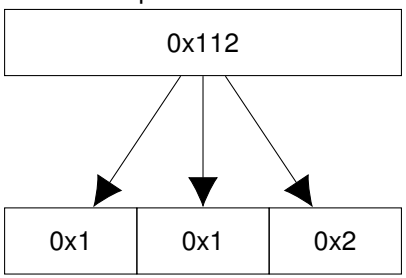
en découpant l'adresse en ses différents champs, on obtient :

- Dossier : 1
- Table : 1
- Décalage : 2

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00		
0x01		
0x02		
0x03		
0x04	0x0A	Folder
0x05	0x0E	
0x06	0x12	
0x07	0x16	
0x08		
0x09		
0x0A	0xC2	Page Table
0x0B	0xE4	
0x0C		
0x0D		
0x0E	0x50	Page Table
0x0F	0x6F	
0x10	0xAC	
0x11		
0x12	0x5F	Page Table
0x13	0x7D	
0x14		
0x15		
0x16	0xF0	Page Table
0x17	0x6A	
0x18	0x8D	
0x19		
0x1A		
0x1B		
0x1C		
0x1D		
0x1E		
0x1F		

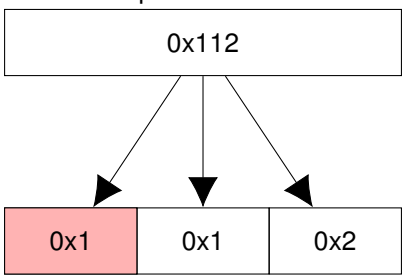
Étape 1

A l'aide du registre CR3, la MMU sait où se situe le dossier contenant un ensemble de tables de pages du processus.

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00		
0x01		
0x02		
0x03		
0x04	0x0A	Folder
0x05	0x0E	
0x06	0x12	
0x07	0x16	
0x08		
0x09		
0x0A	0xC2	Page Table
0x0B	0xE4	
0x0C		
0x0D		
0x0E	0x50	Page Table
0x0F	0x6F	
0x10	0xAC	
0x11		
0x12	0x5F	Page Table
0x13	0x7D	
0x14		
0x15		
0x16	0xF0	Page Table
0x17	0x6A	
0x18	0x8D	
0x19		
0x1A		
0x1B		
0x1C		
0x1D		
0x1E		
0x1F		

Étape 1

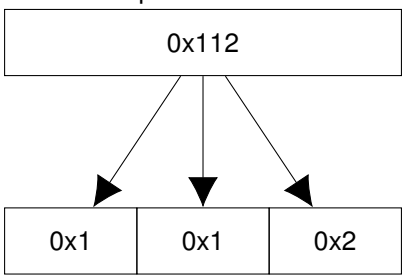
A l'aide du registre CR3, la MMU sait où se situe le dossier contenant un ensemble de tables de pages du processus.

En regardant l'adresse virtuelle, c'est la 2ème table qui est demandée. La MMU récupère ainsi l'adresse de la table, qui est 0x0E.

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00		
0x01		
0x02		
0x03		
0x04	0x0A	
0x05	0x0E	Folder
0x06	0x12	
0x07	0x16	
0x08		
0x09		
0x0A	0xC2	Page Table
0x0B	0xE4	
0x0C		
0x0D		
0x0E	0x50	Page Table
0x0F	0x6F	
0x10	0xAC	
0x11		
0x12	0x5F	Page Table
0x13	0x7D	
0x14		
0x15		
0x16	0xF0	Page Table
0x17	0x6A	
0x18	0x8D	
0x19		
0x1A		
0x1B		
0x1C		
0x1D		
0x1E		
0x1F		

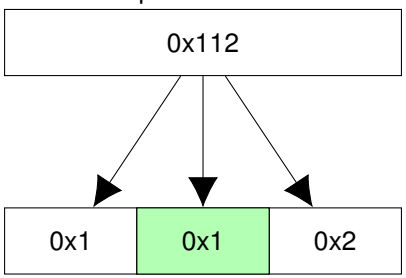
Étape 2

On réitère le même procédé pour récupérer la page demandée, cette fois-ci en partant de l'adresse de la table des pages récupérée précédemment.

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00		
0x01		
0x02		
0x03		
0x04	0x0A	
0x05	0x0E	Folder
0x06	0x12	
0x07	0x16	
0x08		
0x09		
0x0A	0xC2	Page Table
0x0B	0xE4	
0x0C		
0x0D		
0x0E	0x50	
0x0F	0x6F	Page Table
0x10	0xAC	
0x11		
0x12	0x5F	Page Table
0x13	0x7D	
0x14		
0x15		
0x16	0xF0	Page Table
0x17	0x6A	
0x18	0x8D	
0x19		
0x1A		
0x1B		
0x1C		
0x1D		
0x1E		
0x1F		

Étape 2

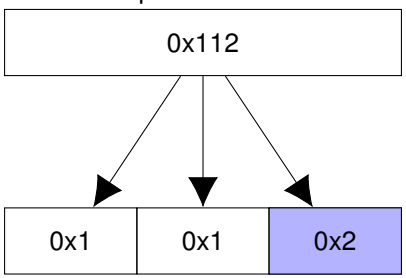
On réitère le même procédé pour récupérer la page demandée, cette fois-ci en partant de l'adresse de la table des pages récupérée précédemment.

En regardant l'adresse virtuelle, c'est la $2^{ème}$ page qui est demandée. La MMU récupère ainsi l'adresse de la page, qui est 0x6F.

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00		
0x01		
0x02		
0x03		
0x04	0x0A	Folder
0x05	0x0E	
0x06	0x12	
0x07	0x16	
0x08		
0x09		
0x0A	0xC2	Page Table
0x0B	0xE4	
0x0C		
0x0D		
0x0E	0x50	
0x0F	0x6F	Page Table
0x10	0xAC	
0x11		
0x12	0x5F	Page Table
0x13	0x7D	
0x14		
0x15		
0x16	0xF0	Page Table
0x17	0x6A	
0x18	0x8D	
0x19		
0x1A		
0x1B		
0x1C		
0x1D		
0x1E		
0x1F		

Étape 3

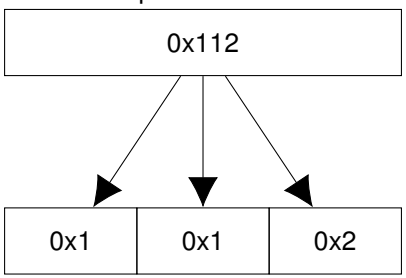
La MMU maintenant connaît l'adresse physique de la page demandée par le processus (0x6F). Il suffit maintenant d'ajouter le décalage pour obtenir l'adresse physique. Le décalage étant de 2, l'adresse finale est 0x6F + 2 = 0x71.

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

Question

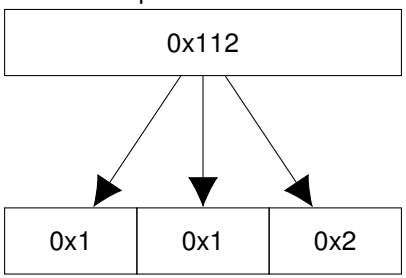
Combien d'accès mémoire est-il nécessaire pour récupérer une donnée ?

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

Question

Combien d'accès mémoire est-il nécessaire pour récupérer une donnée ?

Réponse

- 1 accès pour récupérer l'adresse de la table de pages.
- 1 accès pour récupérer l'adresse de la page.
- 1 accès pour lire la donnée.

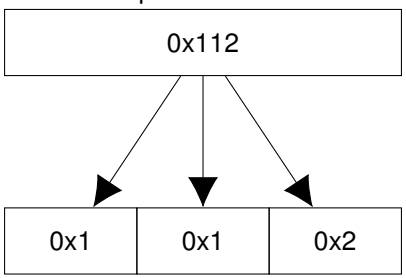
Il faut donc 3 accès en tout.

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

Question

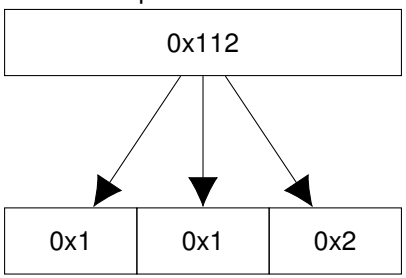
A votre avis, quelle est la taille réelle de chaque champ ?

Exemple sous Linux

Structure d'une adresse virtuelle

Dossier	Table	Decalage
4 bits	4 bits	4 bits

Exemple de traduction



CR3: 0x04

Adresse	Donnée	Info
0x00 0x01 0x02 0x03		
0x04 0x05 0x06 0x07	0x0A 0x0E 0x12 0x16	Folder
0x08 0x09		
0x0A 0x0B	0xC2 0xE4	Page Table
0x0C 0x0D		
0x0E 0x0F 0x10	0x50 0x6F 0xAC	Page Table
0x11		
0x12 0x13	0x5F 0x7D	Page Table
0x14 0x15		
0x16 0x17 0x18	0xF0 0x6A 0x8D	Page Table
0x19 0x1A 0x1B 0x1C 0x1D 0x1E		
0x1F		

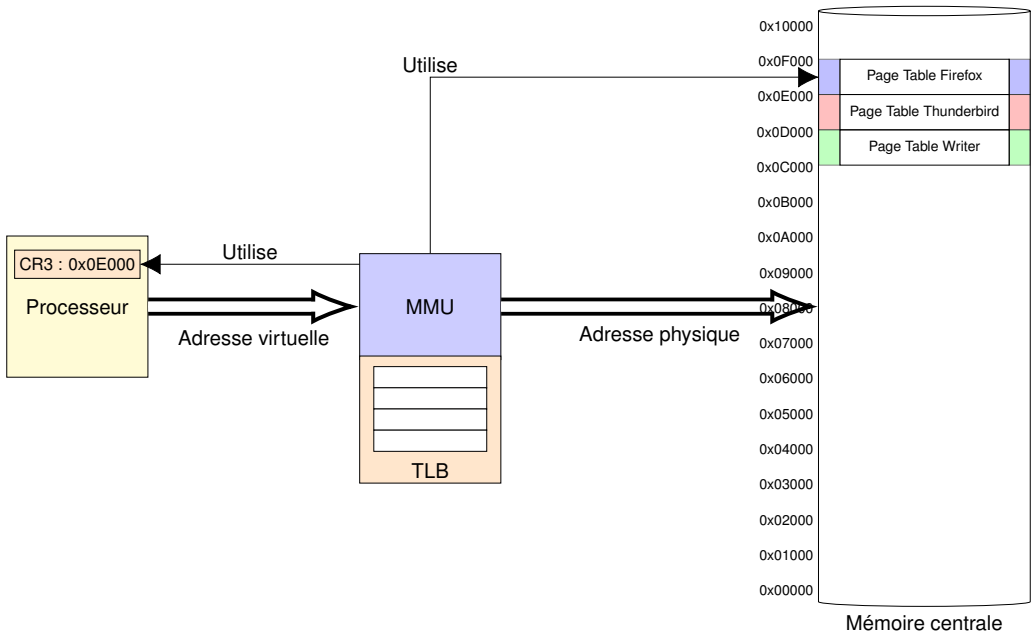
Question

A votre avis, quelle est la taille réelle de chaque champ ?

Réponse

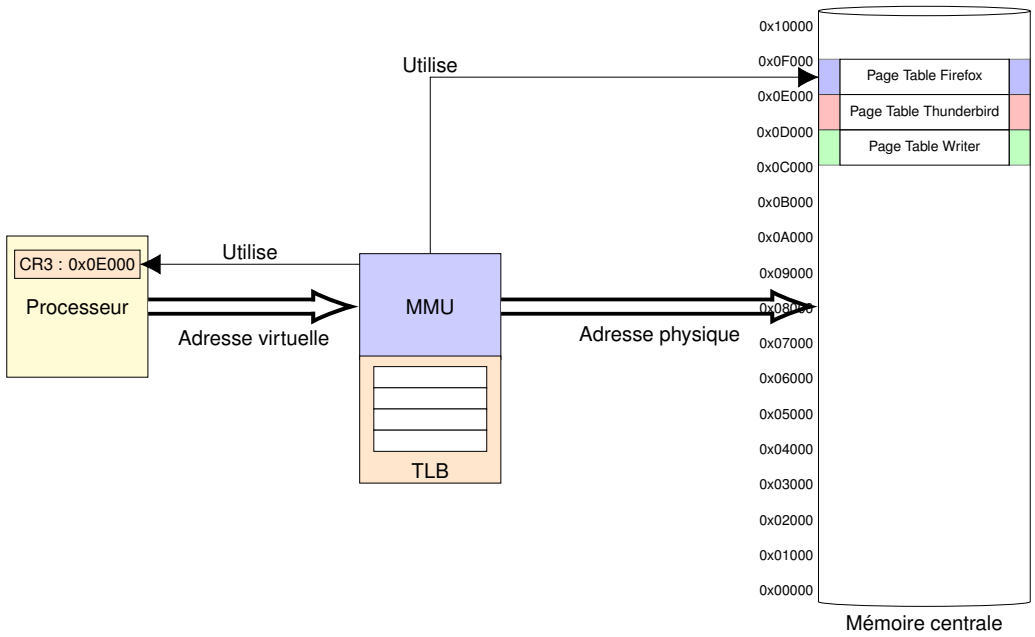
En principe ils sont sur 12 bits, pour coller à la taille d'une page. Un champ plus grand aurait amené à un risque de vulnérabilité, car on pourrait demander une adresse sur le cadre de page adjacent.

Comment accélérer cette traduction ?



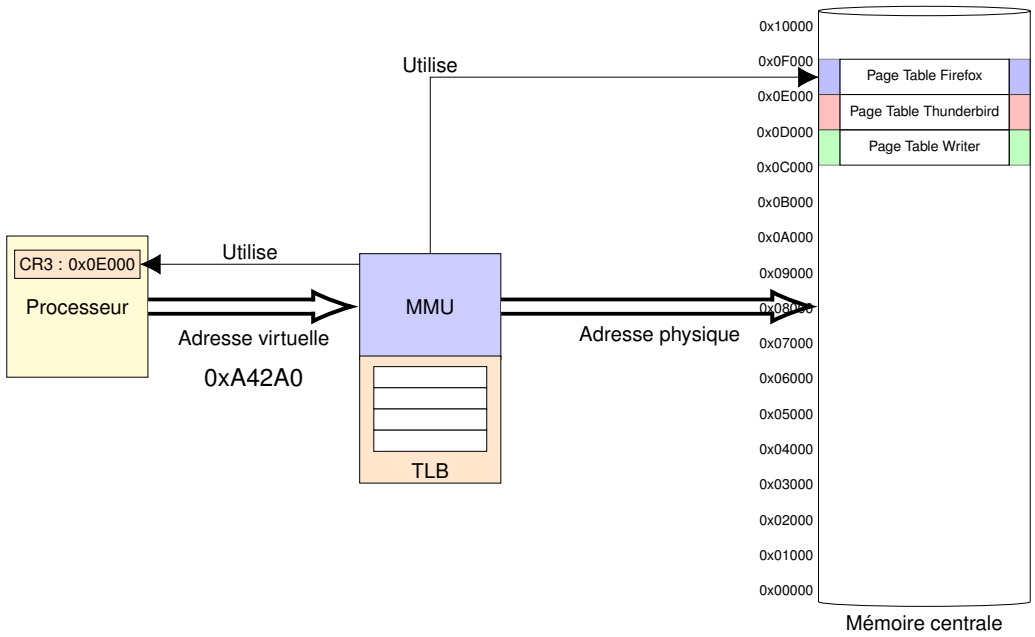
La Translation Lookaside Buffer (TLB)

La TLB est un composant interne de la MMU. C'est un cache, qui garde en mémoire les dernières traductions adresse virtuelle/adresse physique effectuées.



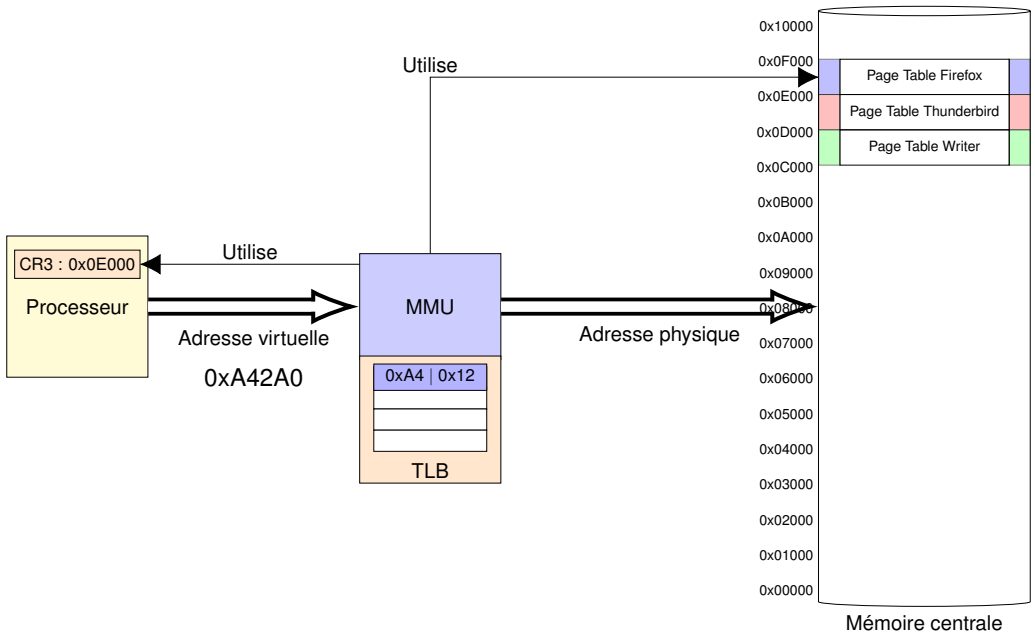
Exemple avec Firefox

Supposons que Firefox s'exécute et demande successivement les adresses 0xA42A0, 0x027DB et 0xA4E06.



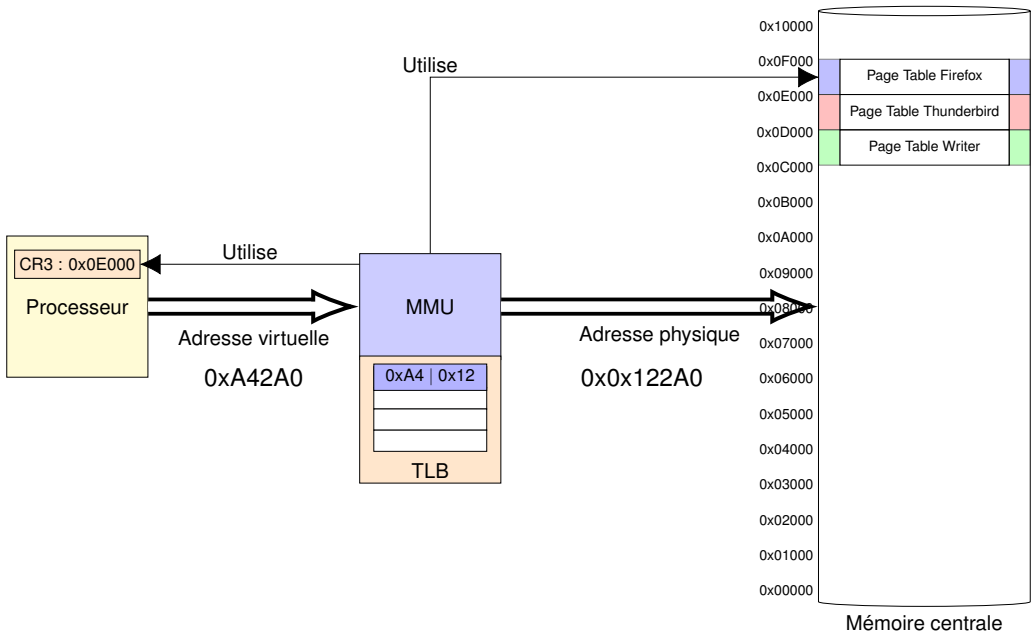
Firefox - Traduction de 0xA42A0

Tout d'abord, Firefox demande l'adresse virtuelle 0xA42A0.



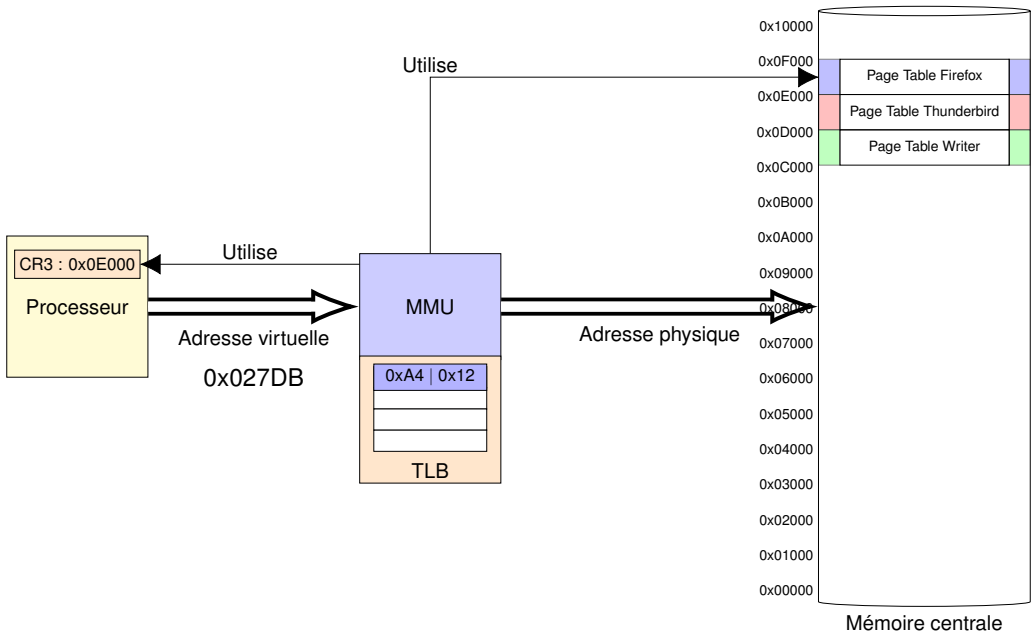
Firefox - Traduction de 0xA42A0

Le TLB étant vide, la MMU doit faire la traduction complète via la table des pages. Après traduction, elle détermine qu'il s'agit du cadre de page 0x12. Elle stocke cette traduction dans son TLB.



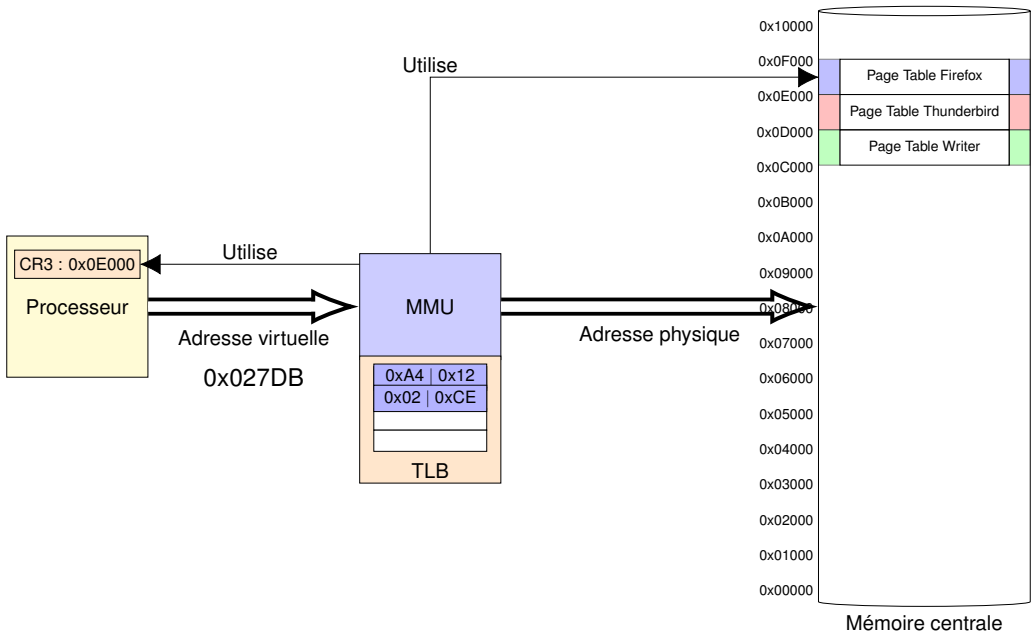
Firefox - Traduction de 0xA42A0

La MMU peut ensuite accéder à l'adresse demandée par Firefox.



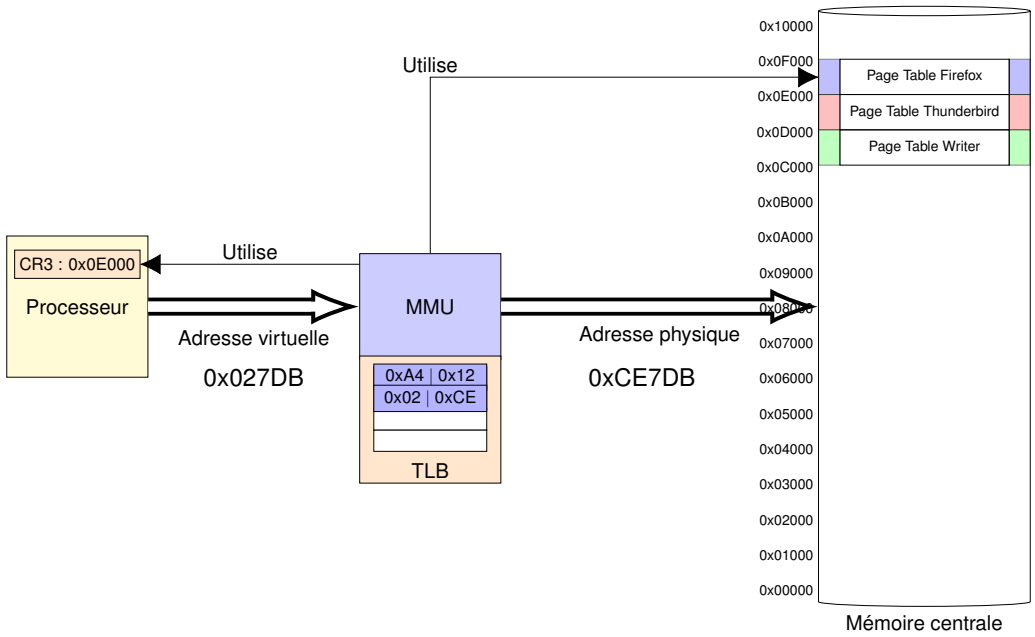
Firefox - Traduction de 0x027DB

Lorsque Firefox demande l'adresse 0x027DB, il demande donc l'octet 0x7BD de la page virtuelle 0x02. Cette page n'est pas dans la TLB, la MMU va donc faire la traduction complète, puis stocker dans la TLB cette traduction (ici 0xCE).



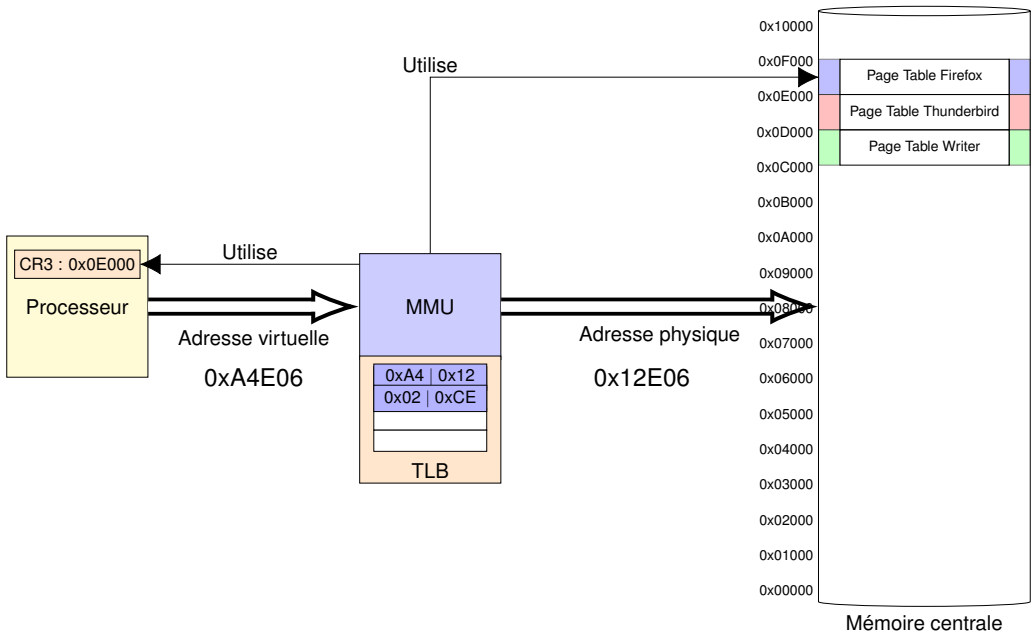
Firefox - Traduction de 0x027DB

Lorsque Firefox demande l'adresse 0x027DB, il demande donc l'octet 0x7BD de la page virtuelle 0x02. Cette page n'est pas dans la TLB, la MMU va donc faire la traduction complète, puis stocker dans la TLB cette traduction (ici 0xCE).



Firefox - Traduction de 0x027DB

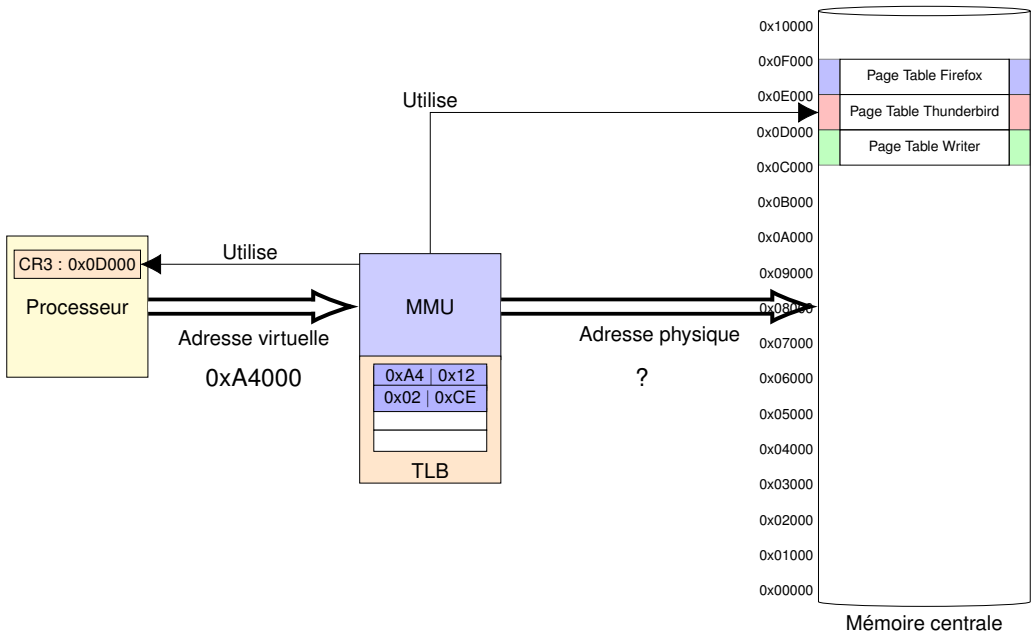
Lorsque Firefox demande l'adresse 0x027DB, il demande donc l'octet 0x7BD de la page virtuelle 0x02. Cette page n'est pas dans la TLB, la MMU va donc faire la traduction complète, puis stocker dans la TLB cette traduction (ici 0xCE).



Firefox - Traduction de 0xA4E06

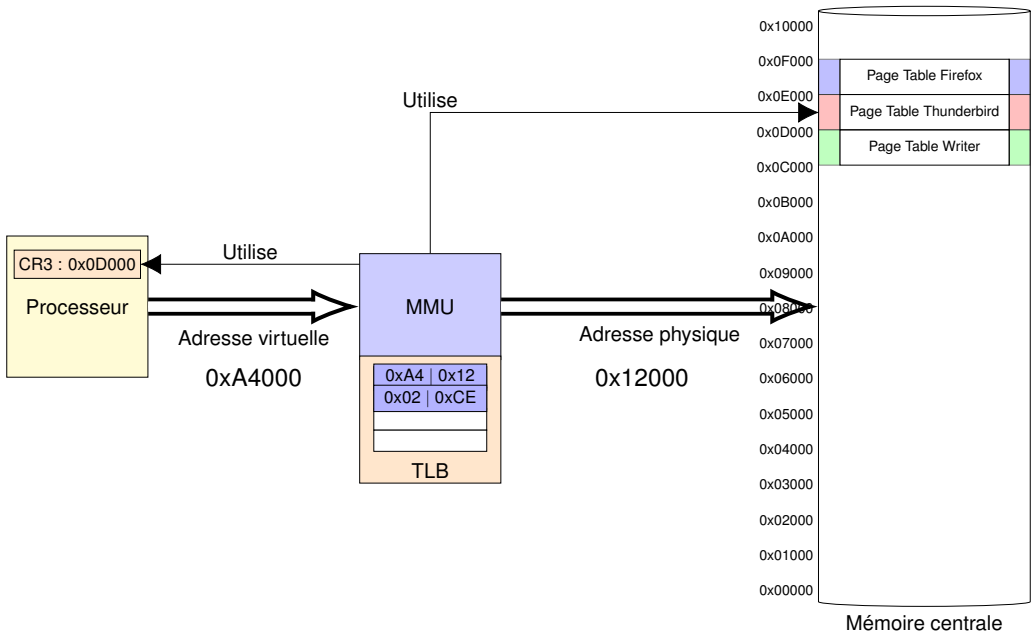
Lorsque Firefox demande l'adresse 0xA4E06, il demande donc l'octet 0xE06 de la page virtuelle 0xA4. La page 0xA4 a déjà été traduite, la traduction est donc immédiate par la MMU.

Mécanisme matériel associé



Question

Et que se passe-t-il si Thunderbird s'exécute et demande l'adresse 0xA4000 ?

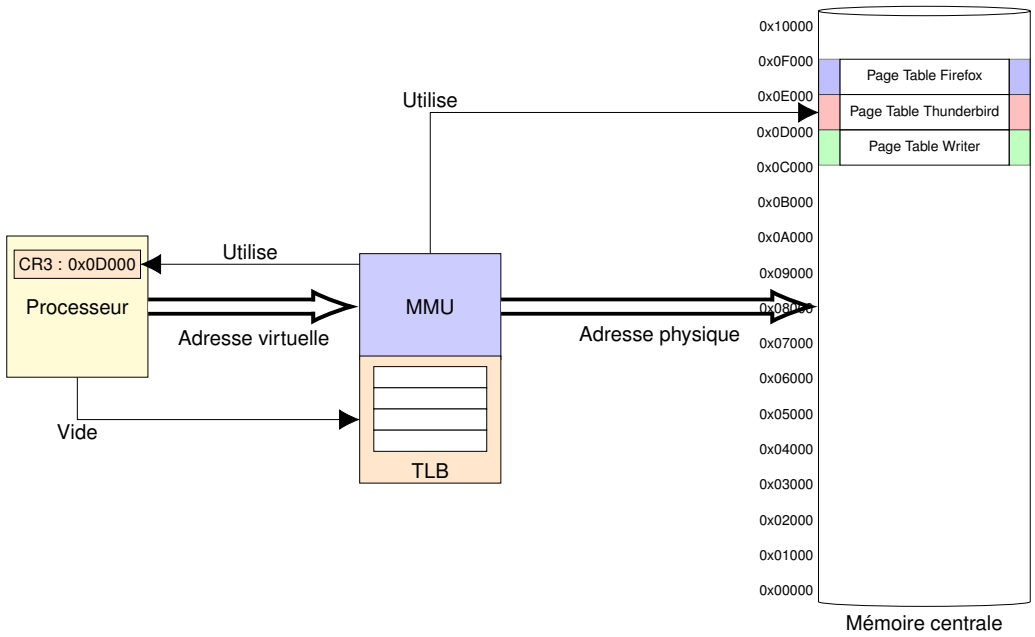


Question

Et que se passe-t-il si Thunderbird s'exécute et demande l'adresse 0xA4000 ?

Réponse

Comme cette correspondance est présente dans son TLB, la MMU va faire la traduction comme s'il s'agissait d'une adresse virtuelle de Firefox !

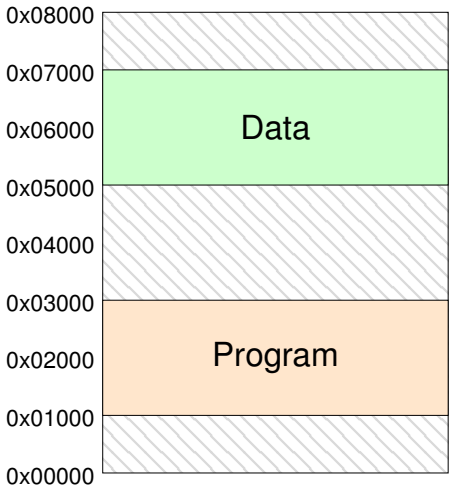


Conclusion

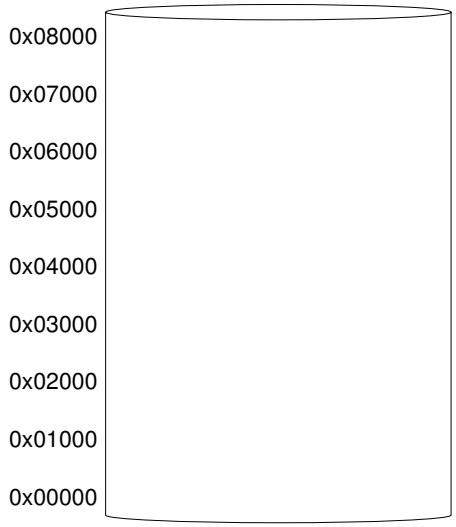
Le processeur doit vider la TLB lors du changement de processus.

Comment optimiser l'utilisation des ressources mémoires ?

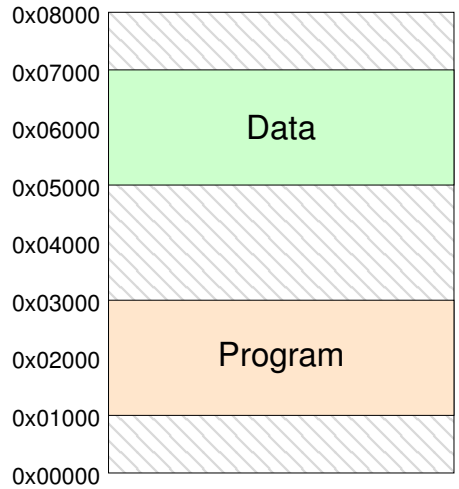
Lancement d'une application 2 fois



Mémoire virtuelle
Processus A



Mémoire centrale

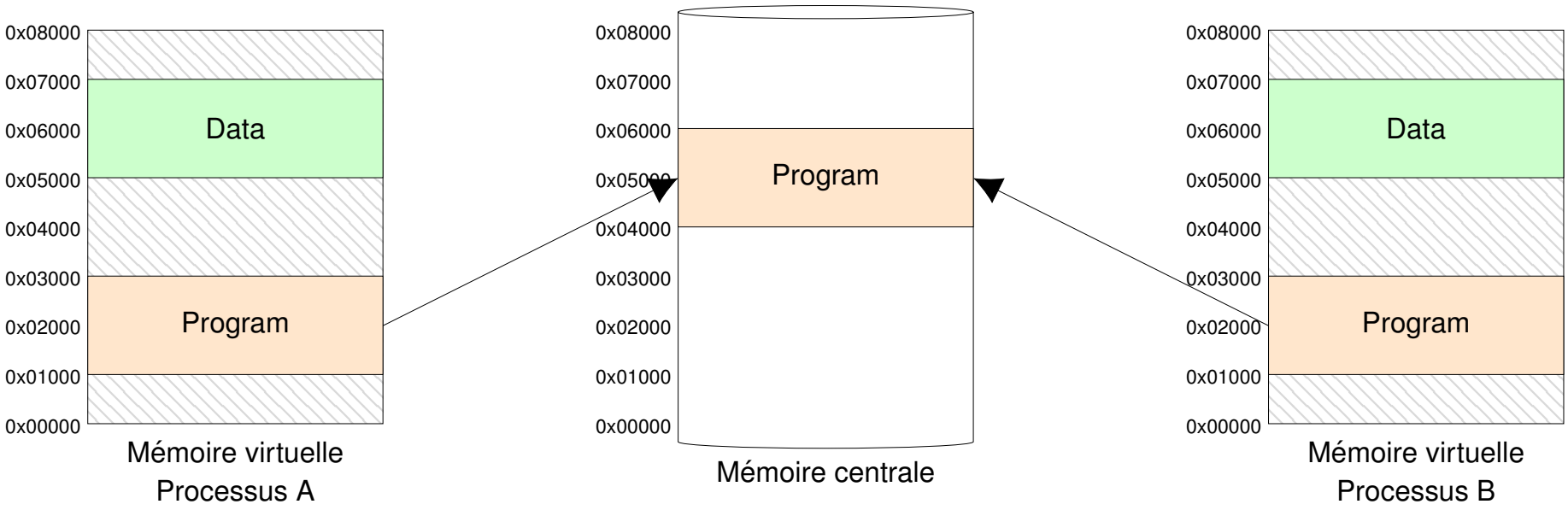


Mémoire virtuelle
Processus B

Question

On suppose que vous lancez deux fois la même application, a-t-on deux programmes distincts en mémoire central ?

Lancement d'une application 2 fois



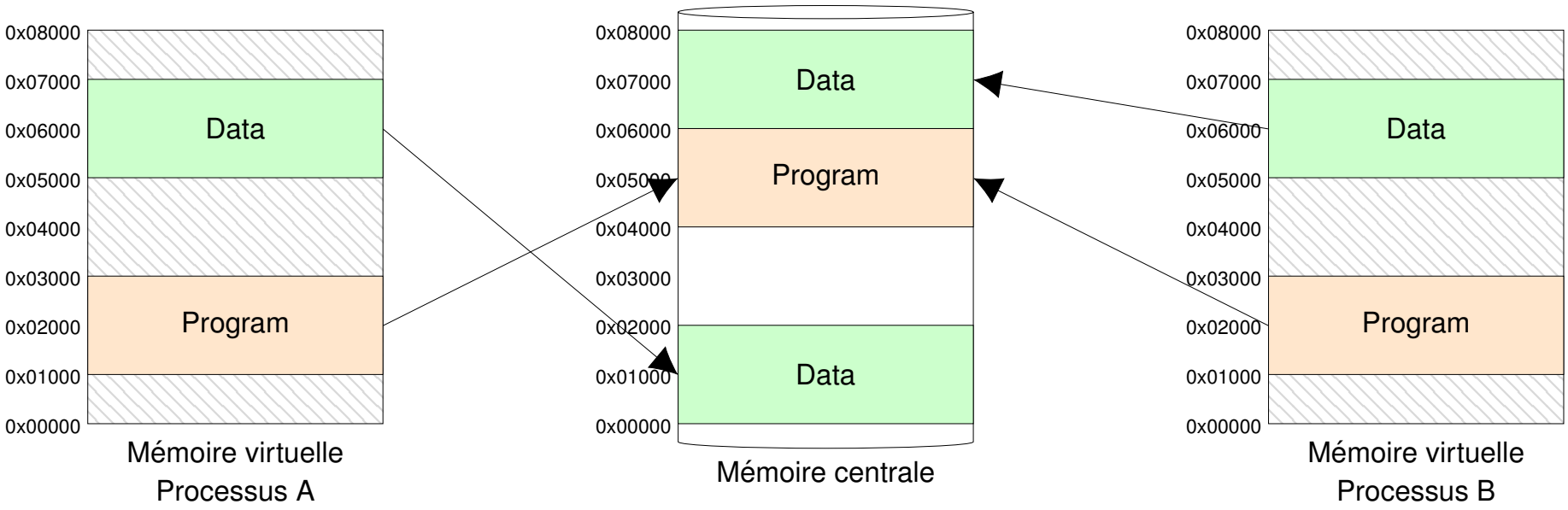
Question

On suppose que vous lancez deux fois la même application, a-t-on deux programmes distincts en mémoire central ?

Réponse

Tant que l'accès à la page se fait en lecture uniquement, il n'y a aucune vulnérabilité apparente. On peut donc avoir 1 seul programme.

Lancement d'une application 2 fois



Et pour les données

Puisque ce sont deux programmes tournant en parallèle, on ne mutualise pas les données, sinon nous aurions deux applications strictement identiques !

Pour aller plus loin

Question

A droite est représenté une partie de l'espace d'adressage virtuel de la même application lancée deux fois. Pourquoi les adresses virtuelles sont différentes, notamment de la pile ?

00007f91acf3e000	4K	r-	ld-2.23.so
00007f91acf3f000	4K	rw-	ld-2.23.so
00007f91acf40000	4K	rw-	[anon]
00007f**ffec9ff**000	132K	rw-	[pile]
00007fffecb6a000	12K	r--	[anon]
00007fffecb6d000	8K	r-x	[anon]

00007fd1b1cc0000	4K	r-	ld-2.23.so
00007fd1b1cc1000	4K	rw-	ld-2.23.so
00007fd1b1cc2000	4K	rw-	[anon]
00007f**feb4eb4**000	132K	rw-	[pile]
00007ffeb4f57000	12K	r--	[anon]
00007ffeb4f5a000	8K	r-x	[anon]

Question

A droite est représenté une partie de l'espace d'adressage virtuel de la même application lancée deux fois. Pourquoi les adresses virtuelles sont différentes, notamment de la pile ?

Réponse

C'est pour éviter les attaques à distance. Si les adresses virtuelles étaient fixées, alors en cas d'injection de code malveillant, l'attaquant saurait précisément quelles adresses attaquer.

00007f91acf3e000	4K	r-	ld-2.23.so
00007f91acf3f000	4K	rw-	ld-2.23.so
00007f91acf40000	4K	rw-	[anon]
00007f**ffec9ff**000	132K	rw-	[pile]
00007fffecb6a000	12K	r--	[anon]
00007fffecb6d000	8K	r-x	[anon]

00007fd1b1cc0000	4K	r-	ld-2.23.so
00007fd1b1cc1000	4K	rw-	ld-2.23.so
00007fd1b1cc2000	4K	rw-	[anon]
00007f**feb4eb4**000	132K	rw-	[pile]
00007ffeb4f57000	12K	r--	[anon]
00007ffeb4f5a000	8K	r-x	[anon]

Remarques complémentaires [1]

Chaque ligne du tableau représente ce que l'on appelle un segment. Il s'agit d'un ensemble de pages contiguës en mémoire virtuelle (mais pas nécessairement contiguës en mémoire centrale) partageant les mêmes droits d'accès.

00007f91acf3e000	4K	r-	ld-2.23.so
00007f91acf3f000	4K	rw-	ld-2.23.so
00007f91acf40000	4K	rw-	[anon]
00007f**ffec9ff**000	132K	rw-	[pile]
00007fffecb6a000	12K	r--	[anon]
00007fffecb6d000	8K	r-x	[anon]

00007fd1b1cc0000	4K	r-	ld-2.23.so
00007fd1b1cc1000	4K	rw-	ld-2.23.so
00007fd1b1cc2000	4K	rw-	[anon]
00007f**feb4eb4**000	132K	rw-	[pile]
00007ffeb4f57000	12K	r--	[anon]
00007ffeb4f5a000	8K	r-x	[anon]

Remarques complémentaires [2]

Il existe 3 droits d'accès :

- r : Accès en lecture autorisé.
- w : Accès en écriture autorisé.
- x : Accès en exécution autorisé.

00007f91acf3e000	4K	r-	ld-2.23.so
00007f91acf3f000	4K	rw-	ld-2.23.so
00007f91acf40000	4K	rw-	[anon]
00007f**ffec9ff**000	132K	rw-	[pile]
00007fffecb6a000	12K	r--	[anon]
00007fffecb6d000	8K	r-x	[anon]

00007fd1b1cc0000	4K	r-	ld-2.23.so
00007fd1b1cc1000	4K	rw-	ld-2.23.so
00007fd1b1cc2000	4K	rw-	[anon]
00007f**feb4eb4**000	132K	rw-	[pile]
00007ffeb4f57000	12K	r--	[anon]
00007ffeb4f5a000	8K	r-x	[anon]

Remarques complémentaires [3]

L'extension .so représente sous Linux une bibliothèque dynamique. Ce sont des fichiers qui contiennent un ensemble de fonctions compilées prêt à l'emploi par le processus. Ce fichier est copié dans l'espace d'adressage virtuel d'un processus à la demande (on dit que le fichier est *mappé*).

00007f91acf3e000	4K	r-	ld-2.23.so
00007f91acf3f000	4K	rw-	ld-2.23.so
00007f91acf40000	4K	rw-	[anon]
00007f**ffec9ff**000	132K	rw-	[pile]
00007fffecb6a000	12K	r--	[anon]
00007fffecb6d000	8K	r-x	[anon]

00007fd1b1cc0000	4K	r-	ld-2.23.so
00007fd1b1cc1000	4K	rw-	ld-2.23.so
00007fd1b1cc2000	4K	rw-	[anon]
00007f**feb4eb4**000	132K	rw-	[pile]
00007ffeb4f57000	12K	r--	[anon]
00007ffeb4f5a000	8K	r-x	[anon]