QCM Moodle IF-3-SYS de contrôle continu, avril 2023

On s'intéresse à un système avec mémoire virtuelle paginée. Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- [x] En temps normal, le mécanisme de traduction d'adresses est invisible pour le programmeur d'application. ✔
- [ ] Le rôle de la MMU est (entre autres) de décider à quelles adresses physiques seront placées les données des processus.
- [x] Un programme qui ne respecterait pas le principe de localité des accès resterait compatible avec la mémoire virtuelle, mais il aurait de mauvaises performances à l'exécution. ✔
- [ ] Lors de chaque accès mémoire d'un processus, le noyau consulte la table de pagination de ce processus pour valider (ou non) la requête.

On s'intéresse dans cette question à un système avec mémoire virtuelle: les adresses virtuelles sont encodées sur 32 bits, et la taille des pages est de 1Mio (c.à.d. 2^{20} octets). Quelle est la taille maximale (en nombre de pages) d'un processus ?

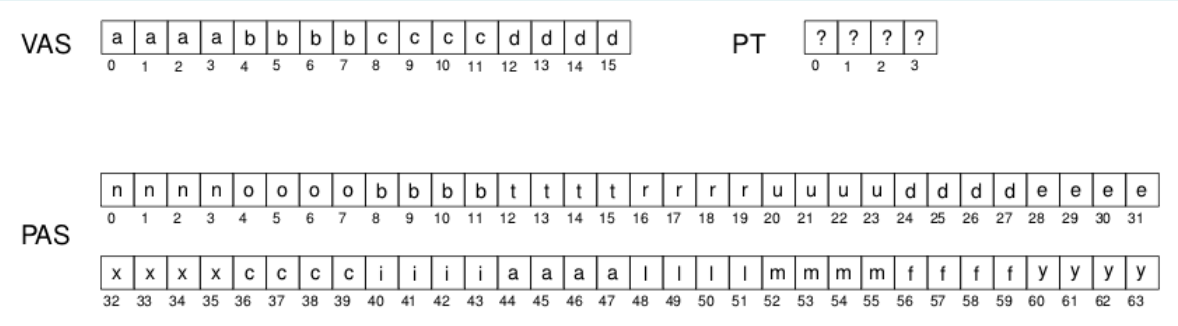
Réponse : 4096 ✔

On s'intéresse à un processus qui vient de causer une "segmentation fault". Pour chaque affirmation ci-dessous, indiquez si elle est compatible avec ces hypothèses:

- [x] Ce processus essayait d'accéder à la mémoire, mais en utilisant une mauvaise adresse virtuelle. ✔
- [ ] Ce processus pourra reprendre son exécution une fois que le noyau aura traité cet évènement.
- [ ] Ce processus essayait de faire un appel système `mmap()`
- [ ] Ce processus essayait d'accéder à la mémoire, mais en utilisant une mauvaise adresse physique.

Le diagramme ci-dessous illustre le VAS d'un processus (16 octets) ainsi que le contenu de la mémoire principale (64 octets). La taille de page est de 4 octets. Remplissez la table de pagination de ce processus (i.e. les points d'interrogation) pour que la traduction d'adresses soit faite correctement:

PT 0: 11 ✔ 1: 2 ✔ 2: 9 ✔ 3: 6 ✔



Dans le TP sur la mémoire virtuelle, on a utilisé `mmap()` pour réimplémenter la commande `cat`. On suppose ici un processus P exécutant le programme `cat-mmap` sur un fichier F. Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- [x] Avec l'appel système `mmap()`, P peut demander au noyau de lui "projeter" le contenu de F dans son VAS. ✔
- [ ] L'appel système `mmap()` renvoie l'adresse physique du contenu de F, ce qui permet ensuite à P de lire ces données directement.
- [ ] Chaque fois que P accède à un octet de F (en lecture ou en écriture), le noyau envoie une requête (de lecture ou d'écriture) au périphérique concerné (typiquement, le disque dur)
- [x] À chaque instant, le contenu de F sera partiellement copié en RAM, mais P n'a pas de moyen pour contrôler finement quelles données seront copiées à quel moment. ✔

Soit la déclaration suivante:

```
int * x = & y ;
```

Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- [ ] La variable y est un pointeur qui contient la valeur de x
- [ ] La variable y est un pointeur qui contient l'adresse de x
- [ ] La variable x est un pointeur qui contient la valeur de y
- [ ] Les variables x et y sont égales.
- [x] La variable x est un pointeur qui contient l'adresse de y ✔

Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- [x] On parle de "défragmentation" mémoire lorsque l'allocateur fusionne plusieurs blocs libres en un seul bloc plus grand. ✔
- [ ] On parle de "fragmentation" mémoire lorsque l'allocateur découpe une requête d'allocation en plusieurs blocs plus petits.
- [x] Certains appels à `malloc()` causent un appel à `mmap()`. ✔
- [ ] Certains appels à `free()` causent un appel à `mmap()`.

Quelle est la valeur affichée par le programme suivant ? On suppose qu'un `float` occupe quatre octets.

```
main()
{
    float v[] = { 3.14, 1.57, 0.785, 0.393, 0.197, 0.0982, 0.0491 }
    float *p = &v[1]
    float *q = &v[3]
    *p = *p + 1
     q =  q - 1
    printf("%d\n",q-p)
}
```

Réponse : 1 ✔

On suppose dans cette question un CPU calculant sur 4 bits: tous les nombres sont représentés sur 4 bits, les calculs sont tronqués à 4 bits, etc. Pour chacune des égalités ci-dessous, indiquez si elle est vraie.

- [ ] `0xA & ~0xC == 0xA - ~0xC`
- [x] `8|3 == 8+3` ✔
- [x] `~2 == -3` ✔
- [x] `~(0xB<<1) == 0xF - (0xF>>1) + 1` ✔

Pour chaque proposition ci-dessous, indiquez s'il s'agit d'un programme correct.

- [ ]
```
int * f (void)
{
  int x = 2023;
  return & x;
}
```

- [ ]
```
int * f (void)
{
  int * px = malloc (sizeof(int));
  px = 2023;
  return px;
}
```

- [x] ✔
```
int * f (void)
{
  int * px = malloc (sizeof(int));
  * px = 2023;
  return px;
}
```

- [ ]
```
int * f (void)
{
  int * px;
  * px = 2023;
  return px;
}
```