IF-3-SYS : Examen du 25 mai 2021

Q1. Vrai (par exemple juste après un fork), Faux (par définition de processus), Vrai (par définition de thread), Faux (absurde).

Q2. Vrai, Faux (c'est un processus ordinaire), Faux (il reste le matériel, notamment le *timer*), Vrai (par définition de *changement de contexte*)

Q3.

```
  time 0   2   6   8  12  14  18  20   28
RR q=4 | A | B | A | B | A | B | A |  B |
```

```
time 0    2   3    5   6    8   9   11  12  28
SRTF |  A | B |  A | B |  A | B |  A |   B   |
```

Q4. SRTF court un risque de famine, donc inacceptable dans un contexte «ouvert» (i.e. on ne connait pas à l'avance les tâches)

Q5. Faux ; Vrai ; Faux ; Vrai

Q6. 1) L'allocateur alloue parfois une zone plus grande que celle demandée. Et il n'a pas de moyen simple d'en avertir l'application (surtout que l'appli s'en fiche un peu).
2) Du coup, l'allocateur note la vraie taille directement dans le bloc alloué.

Q7. 1 ; 2 ; 1 ; 1

Q8. Faux ; Vrai ; Faux ; Vrai

Q9. Par exemple

Init : sem A=0 B=0 C=0
T1 : C1 V(A)
T2 : C2 V(B) V(C)
T3 : P(A) P(B) C3
T4 : P(C) C4

ou encore

Init : sem A=0 B=0
T1 : C1 V(A)
T2 : C2 V(B) V(B)
T3 : P(A) P(B) C3
T4 : P(B) C4

Q10. Faux (même en utilisant tous les «blocs indirects», la structure de données n'est pas récursive) ; Vrai (le noyau écrit dans `*buf` les données lues sur le disque) ; Faux (les répertoires sont des fichiers spéciaux) ; Faux (absurde).

Q11. VAS = 256 octets, découpé en pages de 16 octets, donc il y a 16 pages dans un VAS.
donc il y a 16 PTE dans une PT. donc une PT occupe 16 octets.

Q12. PPN = 7 bits donc 2^7 = 128 pages physiques, de 16 octets chacune. soit $2^7 \times 2^4$ = 2048 octets

Q13. lecture = écriture = 2 accès : l'un pour lire la PTE, l'autre pour faire l'opération à la bonne PA

Q14. La PT est (16 entrées) est : `a9 b4 bf c2 ff d7 fe 03 fd 09 10 87 39 7f 6e 05`
VA=0x65 ⇒VPN=6 ⇒PTE = 0xFE = 0b11111110 ⇒invalide
VA=0x0c ⇒VPN=0 ⇒PTE = 0xA9 = 0b10101001 ⇒PPN = 0b1010100 = 0x54 ⇒PA = 0x54c
VA=0x26 ⇒VPN=2 ⇒PTE = 0xbf = 0b10111111 ⇒PPN = 0b1011111 = 0x5F ⇒PA = 0x5F6
VA=0xA8 ⇒VPN=0xA ⇒PTE = 0x10 ⇒invalide

Q15.

```
void* translate_virtual_to_physical( uint8_t* PT, void* VA )
{
  int VPN = VA/16;
  int PO = VA % 16;
  uint8_t PTE = PT[VPN];
  if(PTE & 1) return INVALID_ADDR;
  int PPN = PTE>>1;
  return (void*) (PPN*16)+PO;
}
```

Q16. Oui, c'est possible. Il suffit qu'un même PTE apparaisse dans les PT de deux processus, et ils verront tous les deux la même page physique.

Q17. Non, en l'état on n'a aucun moyen pour marquer une page en lecture seule.
Il faudrait changer le format des PTE pour y inclure un drapeau «writable» et vérifier chaque écriture via la MMU.