IFA-3-SYS : Examen du 20/03/2024

Q1. Scheduler, ISR, exec, context switch

Q2. c'est une *instruction* machine qui provoque une *IRQ*.
on s'en sert pour implémenter les *appels systèmes*, i.e. un saut volontaire dans le *noyau* avec changement de *mode d'exécution* au passage.

Q3.
```
1
2
3
4
5
6
```

Q4. Faux, Vrai Vrai, Faux

Q5. On attend qu'une condition devienne vraie en *tournant dans une boucle* de scrutation. (par opposition avec *l'attente passive*, pendant laquelle l'exécution est suspendue)
C'est mauvais si ça dure trop, parce que ça *consomme des cycles CPU pour rien*. En particulier, c'est aberrant sur un système *monoprocesseur* : si j'ai la main c'est que les autres processus ne l'ont pas, et donc ils ne peuvent donc rien faire pour moi !

Q6.
```
time 0   7  11  16  25
 CPU | A | C | A | B |
```

Q7. Faux, Vrai, Faux, Vrai

Q8. VA 16 bits et VPN 7 bits ⇒ PO 9 bits ⇒ taille de page 512 octets ⇒ besoin de 10 pages

Q9. variable globale ⇒ dans `.data`

Q10. Faux ; Vrai ; Vrai ; Faux

Q11. Faux ; Faux ; Faux ; Vrai

Q12. Faux ; Vrai ; Faux ; Faux

Q13. init : S=1 (simple mutex). A,C : P(S) et B,D : V(S)

Q14. (cf little book of semaphores readers-writers). Rien de fracassant : on garde une mutex côté les rédacteurs, et côté lecteurs on se débrouille pour que ladite mutex soit verrouillée tant qu'il y a un ou plusieurs lecteurs dans leur section critique.

init : Semaphore mutexNBReaders=1, variable nbReaders=0, Semaphore libre=1

A : P(mutexNBReaders)
if(nbReaders == 0) *// c'est que je suis le premier lecteur à rentrer*
then P(libre) *// marquer la ressource comme non-libre pour bloquer les rédacteurs*
end if
nbReaders += 1
V(mutexNBReaders)

B : P(mutexNBReaders)
nbReaders -= 1
if(nbReaders == 0) *// c'est que je suis le dernier lecteur à repartir*

then V(libre) *// marquer la ressource comme libre pour débloquer les rédacteurs*
end if
V(mutexNBReaders)

Pour les rédacteurs c'est beaucoup plus simple puisqu'on parle toujours d'une pure mutex :
C : P(libre) *// prendre le mutex*
`modifier()`
D : V(libre) *//rendre le mutex*

Q15. il est possible pour les lecteurs de monopoliser la ressource, empêchant les rédacteurs de travailler.

Q16. Une possibilité est de rajouter un troisième sémaphore que le rédacteur peut prendre pour empêcher des lecteurs supplémentaires de rentrer :

init : Sem mutexNBReaders=1, variable nbReaders=0, Sem libre=1, Sem toto=1

A : P(toto) *// vérifier qu'on a l'autorisation de rentrer*
V(toto)
P(mutexNBReaders)
... *// la suite est identique à la question précédente*

Et côté rédacteurs, ça donne :
C : P(toto) *//empêcher de nouveaux lecteurs de rentrer*
P(libre) *// prendre le mutex*
`modifier()`
D : V(libre) *//rendre le mutex*
V(toto)

Downey donne une solution similaire et parle de "turnstile" (trad. «portillon»).