IFA-3-SYS : Examen du 15/03/2023

Q1. Vrai ; Faux (c'est un processus) ; Faux (on peut encore recevoir des interruptions matérielles) ; Vrai (par définition)

Q2. Vrai ; Faux (le scénario est plausible) ; Faux (absurde) ; Faux (il faudrait un `wait()` explicite)

Q3. par exemple

```
void chaine(N)
{
   if(N == 0)
      return;
   int r = fork()
   if(r == 0) // child
      chaine(N-1);
}
```

ou encore

```
void chaine(N)
{
   while(N>0)
   {
      int r = fork();
      if(r != 0 ) // parent
          return;
      N--;
   }
}
```

Q4. Vrai ; Vrai ; Faux ; Faux

Q5.

```
time 0    3   8  12  13  15  18  23  27  28  30  31   32  35  37  42  44  45
 CPU |  A | B |   C   | B | A | C |   B   | C |    B   |   C   | C | C |idle
  IO |idle|   A   | idle  | B |   A   | idle  | C |idle| B |   idle    | C |
```

Q6. A est IO-bound, B et C sont CPU-bound

Q7. Faux (plus lent en moyenne, à cause des défauts de cache) ; Faux (le noyau peut utiliser l'espace d'échange en plus de la RAM) ; Vrai (par définition) ; Faux (entre l'espace d'échange et la mémoire)

Q8. `addr` indique l'adresse virtuelle où j'aimerais que le noyau place ma nouvelle zone. Mais on met NULL pour dire «n'importe où».
`length` indique la taille (en octets) de la zone souhaitée. Exemple : 1000000
`prot` spécifie les permissions que je veux associer à cette nouvelle zone : lecture, écriture, exécution. Valeur typique `PROT_READ | PROT_WRITE`
`flags` indique le type de mapping désiré : privé ou partagé, contenu initial vierge ou basé sur un fichier d'échange. Valeur Typique : `MAP_ANON | MAP_PRIVATE`
`fd, offset` indiquent de quel fichier il s'agit (en cas de `MAP_FILE`).

Q9. 32 bits de VA, et 19 bits de PO $\Rightarrow$ 13 bits de VPN $\Rightarrow$ 2^{13} = 8192 PTE.

Q10. FF : X dans C (reste 80), Y dans A, Z dans D
BF : X dans G, Y dans A, Z dans F
WF : X dans C, Y dans D, Z dans H

Q11. `int r=0; while(N) { r += N&1; N >>= 1; }`

Q12. Interblocage (deadlock) si chaque convive prend sa fourchette gauche : maintenant que plus aucune fourchette n'est dispo, tout le monde va devoir attendre éternellement.

Q13. L'attente circulaire n'est plus possible : le dernier philosophe qui voudrait s'asseoir (avant de saisir sa fourchette gauche) doit rester debout, donc l'avant-dernier peut saisir ses deux fourchettes et manger.

Q14. partagé : semaphore chaises=4
saisir : P(chaises) P(gauche) P(droite)
poser : V(gauche) V(droite) V(chaises)

Q15. Quand j'ai faim et que je me lève, il peut m'arriver de :

- obtenir mes deux fourchettes immédiatement, et alors je me rassieds.
- saisir ma fourchette gauche, mais devoir attendre pour la droite. dans ce cas-là je suis sûr de l'avoir tôt ou tard, car mon voisin de droite finira par la reposer. Et pendant ce temps-là, personne d'autre ne peut se lever.
- devoir attendre pour la fourchette de gauche. même argument que pour la droite.

Q16. partagé : mutex debout
saisir : P(debout) P(gauche) P(droite) V(debout)
poser : V(gauche) V(droite)