QUESTIONS CHAPITRE 4

Synchronisation

Question 1

- Qu'est-ce qu'une ressource critique ?

Question 2

- Qu'est-ce qu'une section critique ?

Question 3

- Que se passe-t-il quand plusieurs processus sont en même temps en section critique pour une ressource A.

Question 4

- Donnez une solution au problème de la question 3

Question 5

- Donnez les 3 appels systèmes sont Unix pour manipuler des sémaphores.

Question 6

- Ecrire 2 programmes P1 et P2 permettant à 2 processus d'incrémenter de 1 une ressource critique compte. On utilisera des sémaphores et on écrira le code permettant d'initialiser le sémaphore.

Question 7

- Qu'est-ce qu'un interblocage ?

Question 8

- Quel est le problème des producteurs comsommateurs ?

Question 9

- Donnez une solution au problème des producteurs consommateurs.

Question 10

- Quel est le problème des lecteurs rédacteurs ?

Question 11

- Donnez une solution au problème des lecteurs rédacteurs.

Question 1

- Qu'est-ce qu'une ressource critique ?
- Une ressource est dite critique si des accès concurrents à cette ressource peuvent la mettre dans un état incoherent.

Question 2

- Qu'est-ce qu'une section critique ?
- Une section critique est une partie d'un programme manipulant une ressource critique

Question 3

- Que se passe-t-il quand plusieurs processus sont en même temps en section critique pour une ressource A.
- La ressource peut être dans un état incohérent ou la valeur A peut être différente en fonction des choix de l ordonnanceur

Question 4

- Donnez une solution au problème de la question 3
- Il faut réaliser une exclusion mutuelle : un seul processus peut être en section critique pour une ressource donnée à un moment donné

Question 5

- Donnez les 3 appels systèmes sont Unix pour manipuler des sémaphores.
- sem_init pour initialiser le sémaphore

  sem_wait pour décrémenter le sémaphore : s'il vaut 0 le processs est bloqué

  sem_post pour incrémenter le sémaphore. Si des processus étaient bloqué par un sem_wait, un des processus sera débloqué

Question 6

- Ecrire 2 programmes P1 et P2 permettant à 2 processus d'incrémenter de 1 une ressource critique compte. On utilisera des sémaphores et on écrira le code permettant d'initialiser le sémaphore.

```
int compte;
sem_t s;
sem_init(&s,1,1);
P1 :
sem_wait(&s);
compte=compte+1;
sem_post(&s);
P2 :
sem_wait(&s);
compte=compte+1;
sem_post(&s);
```

Question 7

- Qu'est-ce qu'un interblocage ?
- C'est lorque plusieurs processus sont endormis et attendent un événement par exemple l'incrémentation d'un sémaphore et qu'ils vont rester éternellement bloqués

Question 8

- Quel est le problème des producteurs comsommateurs ?
- Un buffer contient N cases
- Des producteurs produisent des ressouces pour ce buffer. Si le buffer est plein, ils se bloquent.
- Des consommateurs consomment des ressources. Si le buffer est vide, ils sont bloqués.

Question 9

- Donnez une solution au problème des producteurs consommateurs.
- Voir cours

Question 10

- Quel est le problème des lecteurs rédacteurs ?
- Une ressource est partagée par plusieurs processus. Certains processus vont lire cette ressource (lecteur), d'autre vont y écrire (rédacteur).
- Lorsqu'on écrit dans la ressource on ne peut pas la lire à cause des risques d'oncohérences. Il ne peut y avoir qu'un seul rédacteurs à un moment donné.
- Plusieurs processus peuvent lire en même temps la ressource.

Question 11

- Donnez une solution au problème des lecteurs rédacteurs.
- Voir cours

Question 1

Qu'est-ce q'un thread ?

-

Question 2

Quelle est le différence entre un thread et un processus ?

•

Question 3

- Quels sont les avantages d'un thread comparé aux processus ?
•

Question 4

- Quels sont les inconvénients d'un thread comparé aux processus ?
•

Question 5

- Donnez un appel système sous Unix permettant de créer un thread.
.

Question 6

- Ecrire un programme où le thread principal crée 2 threads qui affiche bonjour.

Question 8

- Quel est l'appel système permettant à un thread d'attendre la fin d'un thread qu'il a créé ?

Question 9

- Donnez 3 appels système permettant de créer une exclusion mutuelle sur une variable partagée par plusieurs threads.

Question 1

Qu'est-ce q'un thread ?

- un thread (ou fil d’exécution) est l’exécution d’une procédure/fonction en parallèle de la fonction principale d’un processus

Question 2

Quelle est le différence entre un thread et un processus ?

- Les ressources des différents threads d'un processus sont partagées alors que lorqu'il y a plusieurs processus les ressources de ceux ci ne sont pas partagés

Question 3

- Quels sont les avantages d'un thread comparé aux processus ?
- La commutation entre les threads est plus rapide

Question 4

- Quels sont les inconvénients d'un thread comparé aux processus ?
- C'est au programmeur d'arbitrer les ressources

Question 5

- Donnez un appel système sous Unix permettant de créer un thread.
  int pthread_create(pthread_t* p_tid, pthread_attr_t* attr, void *(*fonction)(void arg), void* arg);
- Le paramètre p_tid spécifie le numéro de l'activité. Elle est de type pthread_t (un entier).
- Le paramètre attr est un paramètre qui définit les attributs de l'activité. On passera en
- générale le pointeur NULL qui donne à l'activité les attributs standards.
- Le paramètre fonction est un pointeur sur la fonction que va exécuter le thread.
- Le paramètre arg est un pointeur sur les arguments de la fonction.

Question 6

- Ecrire un programme où le thread principal créent 2 threads qui affiche bonjour.

```
pthread_t pthread_id[2];
void f_thread()
{printf( « BONJOUR\n » );}
int main()
{int i;
for(i=0;i<2;i++)
    pthread_create(&pthread_id[i],,NULL, (void*) f_thread,NULL)    ;

sleep(3);
}
```

Question 7

- Quel est l'appel système permettant de terminer un thread en renvoyant un code de retour au thread principal.

int pthread_exit (void* p_status);
Cette fonction termine le thread.
Le paramètre p_status correspond à un pointeur sur le code de retour du thread.

Question 8

- Quel est l'appel système permettant à un thread d'attendre la fin d'un thread qu'il a créé ?
  int pthread_join (pthread_t tid, void** status);
- Cette fonction permet à une activité d'attendre la terminaison d'une autre activité et de récupérer le code de retour de cette activité (définit par l'appel pthread_exit()).
- Le paramètre tid est l'identité de l'activité pour laquelle on attend la terminaison.
- Le paramètre status est un pointeur sur un pointeur du code de retour.

Question 9

- Donnez 3 appels système permettant de créer une exclusion mutuelle sur une variable partagée par plusieurs threads.

  pthread_mutex_init()

  pthread_mutex_lock()

  pthread_mutex_unlock()

QUESTION CHAPITRE 6

- Gestion d'un tas.

Question 1

- Comment sont gérées les variables globales d'un programme ?

Question 2

- Comment sont gérés les paramètres et les variables locales d'un appel de fonction ?

Question 3

- Comment sont gérés la mémoire utilisée par un malloc et un free.

Question 4

- Comment est gérée la mémoire occupée par l'exécutable d'un programme.

Question 5

- Que doit faire un tas ?

Question 6

- Quels sont les critères de performances du gestionnaire d'un tas ?

Question 7

- Comment est géré un tas qui utilise une carte de bits ?

Question 8

- Quel est le principal problème d'une carte de bits ?

Question 9

Comment un tas peut-il être gérée par une liste chaînée.

Question 10

- Donnez l'algorithme d'allocation d'un bloc dans un tas géré par une liste chaînée.

Question 11

- Donnez 3 variantes de la questiion 10

Question 12

Quel sont les avantages et les inconvénients de la variante 1

Question 13

Quel sont les avantages et les inconvénients de la variante 2

Question 14

Quel sont les avantages et les inconvénients de la variante 3

Question 15

- Donnez l'algorithme de désallocation d'un bloc d'un tas géré par une liste chaînée

Question 16

- Quel est le principal inconvénient de l'algorithme d'allocation d'un tas géré par un liste chaînée.

Question 17

- Dans un tas géré par la méthode buddy quel est la structure de données des différents blocs de mémoire.

Question 18

- Dans la méthode buddy, quand on veut un bloc de t octets, quels est la taille du bloc réellement alloué ?

Question 19

- Quel est le principaal avantage de la méthode buddy ?

Question 20

- A quoi sert un ramasse miette ?

Question 21

- Qu'est-ce qu'un compteur de références ?

Question 22

- Donnez un exemple complet où un compteur de références est pris en défaut.

Question 1

- Comment sont gérées les variables globales d'un programme ?
- Par de la mémoire allouée statiquement lors du lancement du processus

Question 2

- Comment sont gérés les paramètres et les variables locales d'un appel de fonction ?
- Par la pile système

Question 3

- Comment sont gérés la mémoire utilisée par un malloc et un free.
- Par un tas

Question 4

- Comment est gérée la mémoire occupée par l'exécutable d'un programme.
- Par de la mémoire allouée statiquement lors du lancement du processus

Question 5

- Que doit faire un tas ?
- Il doit gérer une memoire partagée entre les différents processus, il doit être capable d'allouer un bloc d'une taille donnée et de libérer ce bloc

Question 6

- Quels sont les critères de performances du gestionnaire d'un tas ?
  - rapidité des opérations d’allocation et de libération
  - surcoût en mémoire
  - gaspillage (mémoire inutilisée sans raison)

Question 7

- Comment est géré un tas qui utilise une carte de bits ?
- La zone est découpée en blocs de tailles fixe. Un tableau de bits indique pour chaque zone si elle est ligne ou occupée

Question 8

- Quel est le principal problème d'une carte de bits ?
- Si les blocs sont petits, l'allocation est complexe et sa durée dépend de la taille de la zone souhaitée.
  S'ils sont gros, il y a beaucoup de gaspillage

Question 9

Comment un tas peut-il être gérée par une liste chaînée.

- Une liste chainée indique pour chaque bloc s'il est libre ou occupé, sa taille et son propriétaire (processus)

Question 10

- Donnez l'algorithme d'allocation d'un bloc dans un tas géré par une liste chaînée.
- Pour rechercher un bloc de taille t on parcours la liste chaînée à la recherche d'un bloc libre de d'une taille supérieure ou égale à t

Question 11

- Donnez 3 variantes de la questiion 10
  - Premier trouvé (first fit) : méthode la plus simple et la plus rapide. On s'arrête au premier bloc qui convient
  - Meilleur trouvé (best fit) : cherche la zone libre la plus petite, c.à.d. la plus proche de la taille recherchée.
  - Pire trouvé (worst fit) : cherche la zone libre la plus grande.

Question 12

Quel sont les avantages et les inconvénients de la variante 1

First fit est la plus rapide mais on va parfois créer des zones petites incapables d'être allouées ou parfois créer trop de petites zones pénalisant les processus recherchant des grandes zones

Question 13

Quel sont les avantages et les inconvénients de la variante 2

Best fit : on ne va pas scinder un gros bloc lorsqu'un processus veut peut de mémoire permettant ainsi l'allocation de grandes zones mais on va parfois créer beaucoup de toutes petites zones difficilement allouables

Question 14

Quel sont les avantages et les inconvénients de la variante 3

Worst fit : on va scinder beaucoup de grosses zones rendant diffcile l'alocation de beaucoup de mémoire. Par contre il y aura moins de miettes

Question 15

- Donnez l'algorithme de désallocation d'un bloc d'un tas géré par une liste chaînée
- On met le bloc dans l'état libre et on fusionne éventuellement les blocs libres avec celui devant ou derrière

Question 16

- Quel est le principal inconvénient de l'algorithme d'allocation d'un tas géré par un liste chaînée.
- S'il y a N blocs, il faut un temps proportionnel à N pour allouer de la mémoire. Cela peut être long

Question 17

- Dans un tas géré par la méthode buddy quel est la structure de données des différents blocs de mémoire.
- C'est un arbre

Question 18

- Dans la méthode buddy, quand on veut un bloc de t octets, quels est la taille du bloc réellement alloué ?
- Parfois on ve allouer un bloc d'une taille de 2t-1

Question 19

- Quel est le principaal avantage de la méthode buddy ?
- S'il y a N blocs, l'allocation et la libération de la mémoire se fait en un temps proportionnels à log(N) c'est plus rapide

Question 20

- A quoi sert un ramasse miettes ?
- Parfois un programme ne possède plus de pointeur vers une zone mémoire qui lui a été allouée mais non libérée. Le ramasse miettes est chargé de repérer ces zones et de les libérer

Question 21

- Qu'est-ce qu'un compteur de références ?
- C'est un compeur qui compte le nombre de pointeurs vers un élément du tas. S'il est à 0, l'élément est désalloué

Question 22

- Donnez un exemple complet où un compteur de références est pris en défaut.
  Imaginons qu'il y ait 3 zones A, B et C dans le tas
  A possède un pointeur vers B
  B possède un pointeur vers C
  C possède un pointeur vers A
  Si le programme n'a plus un seul pointeur vers A, B ou C, aucun des compeurs de référence n'est à 0

QUESTION CHAPITRE 7

- Ordonnancement

Question 1

- A quoi sert un ordonnanceur ?

Question 2

- Donnez les 3 principaux états d'un processus dans un ordonnaceur en les expliquant.

Question 3

- Quelles sont les 2 qualités importantes d'un ordonnaceur. Expliquez.

Question 4

- Qu'est-ce que la préemption pour un ordonnanceur ?

Question 5

- Dans un ordonnanceur non préemptif, quand un processus passe-t-il de l'état élu à l'état éligible en donnat l'appel système impliqué ?

Question 6

- Comment fonctionne un ordonnanceur FIFO sans préemption ?

Question 7

- Quels sont les avantages et les inconvénients d'un ordonnanceur sans préemption

Question 8

- Commet fonction un ordonnaceur basé sur un tourniquet ?

Question 9

- Quels sont les avantages et les inconvénients d'un ordonnanceur avec préemption

Question 10

- Donnez le principe de base d'un ordonnanceur shortest job next ?

Question 11

- Donnez les avantages et les inconvénients d'un ordonnanceur Shortest job next.

Question 12

- Dans un ordonnaceur qu'est-ce qu'une priorité interne et une priorté externe ?

Question 13

- Sous Unix quelle est la commande permettant de modifier la priorité externe ?

Question 14

- Qu'est-ce qu'un ordonanceur temps réel ?

Question 15

- Donnez un exemple d'utilisation d'un ordonnanceur temps réel ?

Question 1

- A quoi sert un ordonnanceur ?
- C'est la partie du systèe d'exploitation partageant le temps processseur entre les processus

Question 2

- Donnez les 3 principaux états d'un processus dans un ordonnaceur en les expliquant.
- Elu : le processus est en cours d'exécution
- Eligible : le processus peut à tout moment dévenir élu
- Bloqué : le processus attend un événement, par exemple la fin d'un appel système et ne peut pas devenir élu immédiatement

Question 3

- Quelles sont les 2 qualités importantes d'un ordonnaceur. Expliquez.
- Il doit garantir l'équité : chaque processus doit avoir un temps processus identique
- Il doit éviter les famines c'est à dire les situations où un processus ne se trouve jamais dans l'état élu

Question 4

- Qu'est-ce que la préemption pour un ordonnanceur ?
- C'est la capacité qu'à le système d'arrêter un processus s'il le désire

Question 5

- Dans un ordonnanceur non préemptif, quand un processus passe-t-il de l'état élu à l'état éligible en donnat l'appel système impliqué ?
- Le processus doir rendre la main volontairement par l'appel système yield

Question 6

- Comment fonctionne un ordonnanceur FIFO sans préemption ?
- Les processus éligibles sont mis dans une file d'attente premier entré premier sorti. Il y a commutaton de tâche lorsque le processus se termine ou lorsqu'il rend la main par yield ou lors d'un appel système

Question 7

- Quels sont les avantages et les inconvénients d'un ordonnanceur sans préemption
- L'ordonnancement est rapide et peut couteux. Par contre un processus peut bloquer éternellement tout le monde. Diffcile d'avoir l'éqité et pas de famines

Question 8

- Comment fonction un ordonnanceur basé sur un tourniquet ?
- C'est une fifo avec préemption. Chaque processus est dans un file d'attente premier entré premier servi. L'ordonnanceur donne un quota de temps au premeier processus. Celui-ci se termine lorsque le processus est fini, lorsque le quota de temps est épuisé ou lorsqu'il y a un appel système

Question 9

- Quels sont les avantages et les inconvénients d'un ordonnancement tourniquet
- On évite les famines mais on pénalise les processus effectuant beaucoup d'appels système

Question 10

- Donnez le principe de base d'un ordonnanceur shortest job next ?
- Pour déterminer quel processus dévient élu, on sélectionne celui qui a le plus de chance de se terminer rapidement

Question 11

- Donnez les avantages et les inconvénients d'un ordonnanceur Shortest job next.
- On est plus équitable avec les processus faisant beauoup d'entrée sortie mais on risque de créer des famines pour les processus faisant beaucoup d'entrée sortie

Question 12

- Dans un ordonnaceur qu'est-ce qu'une priorité interne et une priorté externe ?
- Un priorité externe est définie par l'utilisateur d'un processus alors qu'une priorité interne st définie par l'utilisateur

Question 13

- Sous Unix quelle est la commande permettant de modifier la priorité externe ?
- nice

Question 14

- Qu'est-ce qu'un ordonnanceur temps réel ?
- C'est un ordonnaceur devant garantir des contraintes de temps lourdes. Par exemple qu'un processus est exécuté au moins une fois par seconde

Question 15

- Donnez un exemple d'utilisation d'un ordonnanceur temps réel ?
- Un ordinateur controlant une usine par exemple une centrale nucléaire