Université Aboubekr BELKAID Tlemcen
Faculté des Sciences
Département d'informatique

Classe LMD S5 Informatique
Année Universitaire : 2020 / 2021

Solution TD N° 1 : Système d'exploitation

Questions de cours :

1- Donner la différence entre un système multi processus, un système multiprocesseur et un Système **distribué** (réparti).

Réponse :

Un système multiprocessus : plusieurs processus s'exécute en parallèles

Un système multiprocesseur : plusieurs processeurs s'exécute en parallèles et partage la mémoire commune (la ram), ils sont reliés entre eux par des bus de données

Un système distribué : c'est un système composé par plusieurs processeurs qui ne partage ni la mémoire ni la fréquence d'horloge et communique entre eux par un réseau de communication.

2- Soit 3 taches A, B et C à faire tourner sur une machine monoprocesseur. Expliquer pourquoi le temps de réponse total est plus grand quand on exécute les trois taches séquentiellement que quand on les exécute parallèlement avec un noyau multitâche (malgré le temps perdu dans la commutation de contexte).

Réponse :

Dans un système séquentiel, un processus qui s'exécute les entrés / sorties ne libère pas le processeur malgré qu'il ne l'utilise pas (il n'y a pas une exécution parallèle entre les traitements cpu et les traitements des entrés/sorties.

Par contre dans un système parallèle lorsque un processus demande des entrés/ sorties libère automatiquement le processeur aux autres processus (il y a une exécution parallèle entre les traitements CPU et les traitements entrés/sortie.

3- Quel est l'appel système dans Unix qui permet de créer un processus fils ?

Réponse :

C'est la fonction fork()

La fonction fork crée un processus fils qui s'exécute en parallèle avec son processus père , le processus fils est une copie de son père au moment de sa création.

Chaque processus fils ou père possède son propre espace mémoire

La fonction fork retourne 3 valeurs :

-1 : échec de création d'un processus

0 dans l'espace mémoire de processus fils

Valeur >0 c'est le pid de processus fils dans l'espace mémoire de processus père

4- Quel est l'intérêt de la notion de processus dans un système d'exploitation ?

Réponse :

C'est la multitâche et la rapidité d'exécution

5- Qu'est ce qu'un thread quel est leur avantage ?

Réponse :

Le thread exprime le parallélisme à l'intérieur d'un processus, il augmente la rapidité d'exécution d'une applications(réduire le temps de réponse d'une application.

6- Pourquoi est-il plus efficace pour une application de créer un seul processus à multi-thread plutôt qu'une application à multi-processus ?

Réponse :

Crée une application multithread est très efficace de créer une application multiprocessus.
Les threads consomment moins de ressources du système par rapport aux processus
Exemple : une application de 3 processus nécessite trois espaces mémoire (chaque processus possède son propre mémoire), par contre une application de 3 threads utilise et partage un seule espace mémoire de son processus père.
Les threads s'exécutent pas des appels systèmes, ils sont inconnue (invisible) au niveau de système.
Touts les threads d'un même processus partage les même ressources de processus père.
La communication entre les threads s'effectué directement en utilisant l'espace mémoire partagé,
Par contre, la communication entre processus nécessite des appels systeme

7- Y a-t-il une limite au nombre de threads crées au sein d'un processus ? si oui laquelle.

Réponse :

Oui, cela dépend de la taille de l'espace mémoire réservé et alloué au processus père

8- Un thread à t-il accès à toutes les ressources présenter sur la machine physique ?

Réponse :

Non, un thread accède uniquement au ressources alloué a son processus père.

9- Soient les deux fonctions C suivantes

```
int tab[10000] ;
void fonction1() {
 int i ;
  for(i=0;i<500;i++) {
    tab[i]=i;
  }
}
```

```
void fonction2() {
 int i ;
  for(i=500;i<10000;i++) {
    tab[i]=i;
  }
}
```

Les cases du tableau tab sont initialisées à zéro. Donnez le contenu du tableau tab après la fin d'exécution des deux fonctions dans les trois cas suivants :

a) Les deux fonctions s'exécutent dans deux processus différents (chaque fonction dans un seul processus chacune).

Réponse :

Fonction 1 →processus1 : tab[1000]={**0,1,2,……499**,0,0……..0}

ou

Fonction 2 →processus2 : tab[1000]={0,0,0**,……0,500,501,502**,……..**999**}

Les deux processus ne partage pas le même tableau tab (chaque processus possede son propre espace mémoire)

b) Les deux fonctions s'exécutent dans deux threads du même processus (chaque fonction dans un seul thread chacune).

Réponse :

Fonction 1 →processus1 : tab[1000]={**0,1,2,……499**,0,0……..0}

Fonction 2 →processus2 : tab[1000]={**0,1,2,……499,500,501,502**,……..**999**}

Donc tab[]={**0,1,2,……499,500,501,502**,……..**999**}

Les deux threads partage le meme tableau tab (meme espace mémoire)

c) Les deux fonctions s'exécutent dans deux threads de deux processus différents (chaque fonction dans un seul thread de chaque processus).

Réponse :

Même résultat que la question a

11- Quelles sont les difficultés principales dans le traitement concurrence d'un système d'exploitation ?

Réponse :

C'est le problème de synchronisation et l'exclusion mutuelle entre les processus

12- à quel moment en utilise l'exclusion mutuelle dans un système d'exploitation ?

Réponse :

Nous utilisons l'exclusion mutuelle lorsque nous avons plusieurs processus qui s'exécutent en parallèles et qui demandent l'accès en même instant a des ressources critiques

13- En générale un système multi-threads nécessite l'utilisation des mécanismes de synchronisation.

Réponse :

OUI, car les threads d'un même processus partage le même espace mémoire de père qui contient des ressources et données partagés.