Projets_10

Tous les projets seront réalisés en Java, ils seront notés avec des notes de 0-10

1. Émuler le fonctionnement des ordonnanceurs en utilisant des threads. L'interface permettra de saisir les paramètres des différents processus en concurrence. L'utilisateur pourra choisir un algorithme d'ordonnancement, parmi les 4 suivants : FCFS, SJF non préemptif, SJF préemptif et Round Robin. Les moments d'arrivée dans la queue des processus prêts ainsi que les durées des rafales saisies par l'utilisateur doivent correspondre aux moments de lancement et durées des threads.

   Le programme affichera graphiquement le déroulement des processus et numériquement les valeurs des temps moyens de traitement et des temps moyens d'attente. Les valeurs comparées des 4 stratégies, pour les mêmes valeurs des paramètres des rafales doivent pouvoir être comparées.

   Réalisation : 4 étudiants (interface commune/ FCFS/SJF non préemptif/SJF préemptif/ Round Robin)

2. Réaliser un système « client/serveur » entre deux machines, permettant de se servir du client pour lancer des opérations de maintenance sur le serveur.

   Réalisation : 2 étudiants (client / serveur)

3. Réaliser un programme de simulation des commandes UNIX (ou Linux) sur une machine équipée de Windows. Établir quelles commandes n'ont pas pu être réalisées et expliquer les raisons.

   Réalisation : 1 étudiant

4. Émuler la gestion de la mémoire virtuelle dans le cas de plusieurs programmes s'exécutant en parallèle sans effectuer des opérations d'E/S. Les programmes seront simulés à l'aide de threads. L'utilisateur devra saisir :

   1. la taille de la mémoire physique disponible
   2. la taille d'une page
   3. le nombre total de programmes a exécuter
   4. le nombre de pages de chaque programme
   5. le nombre de pages chargées au lancement d'un programme
   6. le quota de temps d'exécution en UC que le SE accorde à chaque processus.
   7. le nombre maximum de processus s'exécutant en parallèle

   Si le nombre des programmes à exécuter est supérieur au nombre de processus pouvant s'exécuter en parallèle, les processus en excès seront placés dans une queue d'attente.

   Une interface graphique permettra d'observer le mouvement des pages. Sera affiché la durée d'exécution de chaque programme et la moyenne générale à la fin de la simulation.
   On doit pouvoir étudier l'influence des paramètres de la mémoire physique (taille de la mémoire et de la page) sur les performances.

   Réalisation : 3 étudiants (interface saisie/ interface graphique/ simulation de la gestion)

5. Simuler un système d'interruptions matérielles. La table des interruptions comprendra 4 « colonnes » :
    1. une colonne de codes (1 caractère alphabétique),
    2. une colonne avec les noms des procédures à exécuter
    3. une colonne avec les durées d'exécution (secondes)
    4. une colonne avec les priorités (valeurs entières entre 1 et 16).

    L'utilisateur saisira au clavier des caractères représentant des codes de la table ; les codes saisis démarreront un thread qui exécutera la procédure qui correspond au code. Les procédures ferons toutes la même chose :

    - envoyer un message avec le nom de la procédure et le temps de son déclenchement,
    - exécuter une boucle sans objet d'une durée égale à la valeur spécifiée dans la table
    - à la sortie de la boucle, envoyer un message avec le nom de la procédure et le temps de la fin.

    La boucle devra durer assez longtemps pour que l'utilisateur puisse saisir pendant son déroulement un nouveau code. Si la priorité de la nouvelle procédure est supérieure à celle en cours de déroulement, cette dernière doit s'interrompre et reprendre l'exécution seulement après le traitement de l'interruption prioritaire. L'interruption et la reprise de la procédure interrompue doivent être signalées par des messages

    Une présentation graphique de l'enchaînement des procédures est demandée.

    Réalisation : 2 étudiants (interface/gestion des interruptions)