Formation Automatique et Informatique Industrielle

Master 1 S2

Matière : Systèmes Embarqués et Systèmes Temps Réel SE-STR

Par : ATOUI Hamza

Plan du TD-TP

- Implémentation des algorithmes par MATLAB.
  1. Implémentation des algorithmes d'ordonnancement des **tâches périodiques indépendantes** :
     1. **Algorithme RM.**
     2. **HOME WORK.**
  2. Implémentation des algorithmes d'ordonnancement des **tâches apériodiques indépendantes** :
     1. **Algorithme FCFS.**
     2. **Algorithme SRT.**
     3. **HOME WORK.**

Tâches périodiques Indépendantes

- **Algorithme RM** : fait partie de la famille des algorithmes à **priorité fixe**.
- Le principe consiste à **affecter** aux tâches des **priorités** qui sont **inversement proportionnelles à leurs périodes**.
- Les tâches à ordonnancer doivent être à échéance sur requête Task(r0, C, D=P).

Tâches périodiques Indépendantes

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par RM.
- **Etape 1** : Test de faisabilité/ordonnancement.
- Si l'étape 1 est valide on passe vers les étapes ci-après.
- **Etape 2** : Calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(P1, P2,…).
- **Etape 3** : Détermination de priorité des différentes tâches par : on va associer la plus haute **priorité** à la tâche de la **petite période**.
- **Etape 4** : de tracer les chronogrammes par:
  - De signaler sur long de chronogramme de chaque tâche : {r↑, P↕}.
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

Tâches périodiques Indépendantes

- Algorithme d'ordonnancement par RM :

À chaque TICK_SYSTEM faire

1- Chercher les tâches à l'état READY (chargement de ReadyQueue)

2- Sélectionner la tâche de la plus petite période dans le ReadyQueue.

3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.

4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.

Répéter les 4 étapes jusqu'à la fin de l'intervalle de simulation

Tâches périodiques Indépendantes

- Le TP demande d'implémenter sous MATLAB l'algorithme RM en suivant les étapes ci-après:
- Demander à l'utilisateur de saisir le nombre de tâches qu'on va ordonnancer.
- Demander à l'utilisateur de saisir les paramètres de chaque tâche.
- Ordonnancer les tâches selon l'algorithme RM.
- Afficher les chronos sous la forme d'un message texte et en spécifiant à chaque TICK système l'état de la tache {'S', 'R', 'E'}.
- Afficher le Gantt Chart sous la forme d'un message texte.

Tâches périodiques Indépendantes

- **HOME WORK.**

- A vous mes chers étudiants, implémenter l'algorithme DM et EDF.

Tâches apériodiques Indépendantes

- **FCFS** : est un algorithme se base sur l'exécution de la tâche qui arrive en premier par une **Ready Queue** en **FIFO** (First In First Out).
- La **plus haute priorité** est donnée au tâche qui **arrive en premier**. Si on a plus d'une tâche **arrive au même temps**, dans ce cas, la priorité est l'**ordre** des tâches.

Tâches apériodiques Indépendantes

- Le temps d'allocation du CPU au tâche choisi par FCFS est **coopératif (la tâche prend le CPU jusqu'à la terminaison de C)**.
- Le modèle utilisé par FCFS est : **Task(r0, C, O)**.
  - **r0** : Arrival Date.
  - **C** : Burst Time (Capacity).
  - **O** : Order of task.
- Temps de charge du noyau : Négligeable.
- On n'a pas de délai critique (D = ∞).

Tâches apériodiques Indépendantes

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par FCFS:
- **Etape 1** : Détermination de priorité des différentes tâches par : on va associer la plus haute **priorité** à la tâche qui arrive en premier. Si on a plus d'une tâche **arrive au même temps**, dans ce cas, la priorité est l'**ordre** des tâches.

Tâches apériodiques Indépendantes

- **Etape 2** : de tracer les chronogrammes par :
  - De signaler sur long de chronogramme de chaque tâche : {r↑}.
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

Tâches apériodiques Indépendantes

- Algorithme d'ordonnancement par FCFS :

À chaque TICK_SYSTEM faire

1- Chercher les tâches à l'état READY (chargement de ReadyQueue)

2- Sélectionner la tâche qui arrive en premier dans le ReadyQueue.

3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.

4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.

Répéter les 4 étapes jusqu'à la fin de Burst Time de toutes les tâches

Tâches apériodiques Indépendantes

- Le TP demande d'implémenter sous MATLAB l'algorithme FCFS en suivant les étapes ci-après:
- Demander à l'utilisateur de saisir le nombre de tâches qu'on va ordonnancer.
- Demander à l'utilisateur de saisir les paramètres de chaque tâche.
- Ordonnancer les tâches selon l'algorithme FCFS.
- Afficher les chronos sous la forme d'un message texte et en spécifiant à chaque TICK système l'état de la tache {'S', 'R', 'E'}.
- Afficher le Gantt Chart sous la forme d'un message texte.
- Calculer le AWT et le ATT.

Tâches apériodiques Indépendantes

- **SRT** : est un algorithme se base sur l'exécution de la tâche de la plus petite capacité.
- Si on a plus d'une tâche **a la même capacité et arrive au même temps**, dans ce cas, la priorité est l'**ordre** des tâches.

Tâches apériodiques Indépendantes

- Le temps d'allocation du CPU au tâche choisi par **SRT** est **préemptif**.
- Le modèle utilisé par SJF est : **Task(r0, C, O)**.
  - **r0** : Arrival Date.
  - **C** : Burst Time (Capacity).
  - **O** : Order of task.
- Temps de charge du noyau : Négligeable.
- On n'a pas de délai critique (D = ∞).

Tâches apériodiques Indépendantes

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par SRT:
- **Etape 1** : Détermination de priorité des différentes tâches par : on va associer la plus haute **priorité** à la tâche de la plus petite capacité. Si on a plus d'une tâche **a la même capacité et arrive au même temps**, dans ce cas, la priorité est l'**ordre** des tâches.

Tâches apériodiques Indépendantes

- Algorithme d'ordonnancement par SRT :

À chaque TICK_SYSTEM faire

1- Chercher les tâches à l'état READY (chargement de ReadyQueue)

2- Sélectionner la tâche de la petite capacité dans le ReadyQueue.

3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.

4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.

Répéter les 4 étapes jusqu'à la fin de Burst Time de toutes les tâches

Tâches apériodiques Indépendantes

- Le TP demande d'implémenter sous MATLAB l'algorithme SRT en suivant les étapes ci-après:
- Demander à l'utilisateur de saisir le nombre de tâches qu'on va ordonnancer.
- Demander à l'utilisateur de saisir les paramètres de chaque tâche.
- Ordonnancer les tâches selon l'algorithme SRT.
- Afficher les chronos sous la forme d'un message texte et en spécifiant à chaque TICK système l'état de la tache {'S', 'R', 'E'}.
- Afficher le Gantt Chart sous la forme d'un message texte.
- Calculer le AWT et le ATT.

Tâches apériodiques Indépendantes

- **HOME WORK.**

- A vous mes chers étudiants, implémenter l'algorithme SJF et RR.