M1 ALMA
Université de Nantes
2010-2011

Projet de Travaux pratiques : Systèmes Distribués

MARGUERITE Alain
RINCE Romain

Université de Nantes
2 rue de la Houssinière, BP92208, F-44322 Nantes cedex 03, FRANCE

Encadrant : QUEUDET Audrey

Table des matières

1 Introduction

Introduction : L'objectif est de définir un outil de simulation d'ordonancement de tâche en temps réel. Parmies ses fonctionalités, l'outil devra pour tester les contraintes temporelles d'un ensemble de tâches générées au préalable. La génération de ces tâches entre par dans la conception de l'outil. Cet outil permettra d'exporter le résultat dans un fichier d'extension *.ktr* pour être exploité directement par l'outil graphique Kiwi.

1.1 Format d'entrée

Il est définit dans le langage XML. Sa syntase sera la suivante

1.1.1 Entête

- Des balises `<genTache.AbstractTache-array>` encadrent la totalité du fichier.
- Une tâche périodique sera définie dans une balise `<genTache.TachePeriodique>`
- Une tâche apériodique sera définie dans une balise `<genTache.TacheAPeriodique>`
- Dans une tache tous ses attributs seront définis de la manière suivante `<nom_attribut>valeur_attribut</nom_attribut>`

1.2 Fonctionnement

- Une génération des tâches dans le format définit ci-dessus. Cette génération doit pouvoir être aléatoire ou définie entièrement par l'utlisateur.
- Une analyse d'ordonnançabilité. L'outil affichera à l'utilisateur les résultats des différents test s'afficheront avec les conclusions qui en découlent.
- Un environnement de simulation. L'outil lors du calcul de de l'ordonancement devra afficher les différents événements. Un bilan de ces action sera résumé dans un fichier au terme de l'execution (facultatif).

– Un fichier d'extension *.ktr* sera généré au terme de l'execution, et contiendra le déroulement de l'ordonancement jusqu'a son terme.

2 Génération de tâches dans un fichier

Introduction : Dans le cadre du module Systèmes Distribués, l'opportunité de spécifier et concevoir un générateur de tâches temps réel sdfsdfsdfsdfsdfsdfsdfsdfsdfsdnous est proposé. Le langage d'implémentation étant libre, notre binome à choisi Java. Ce choix est basé principalement par le fait qu'il s'agissait du langage le plus maitrisé. Cela a permis de concentrer nos efforts sur la mise en place des algorithmes et non sur des probèmes de langage. L'objectif est de définir un outil de simulation d'ordonancement de tâche en temps réel. La première partie du projet avait pour objectif d'obtenir un nombre n de taches périodiques et apériodiques pour de futurs traitements décrits dans la partie 2 : rajouter lien dym. Á nouveau le choix du format d'un tel fichier nous était laissé. Nous avons choisis de générer un fichier xml (à nouveau pour des raisons de simplicité) à la syntaxe suivante :

```
<genTache.AbstractTache-array>
  <genTache.TachePeriodique>
    <Pi>377</Pi>
    <ri>0</ri>
    <id>1</id>
    <Ci>1</Ci>
    <Di>1</Di>
  </genTache.TachePeriodique>
  <genTache.TachePeriodique>
    <Pi>162</Pi>
    <ri>0</ri>
    <id>2</id>
    <Ci>6</Ci>
    <Di>30</Di>
  </genTache.TachePeriodique>
  <genTache.TacheAperiodique>
    <ri>859</ri>
    <id>3</id>
    <Ci>26</Ci>
    <Di>71</Di>
```

```
21     </genTache.TacheAperiodique>
22   </genTache.AbstractTache-array>
```

On remarque que les taches périodiques sont identifiées par :

- Pi
- ri
- id
- Ci
- Di

Alors que les tâches apériodiques ont seulement :

- ri
- id
- Ci
- Di

Calcul des tâches ap Le calcul des tâches ap est effectué selon la formule suivante : $U_a = \frac{\sum_{i=1}^{m} C_i}{ppcm(P_i)}$ .... L'utilisateur entre la variable Uap et le nombre de tache ap qu'il désire (m) les variables restantes charge des ap sur une hyperpériode