

**Examen de PaDis**

**Année 2005-2006**

**Master SDRP**

**Le 16 janvier 2006**

**Cours de Serge Chaumette**

Durée 1h30

Documents autorisés : notes de cours et travaux personnels

*Les réponses doivent être rédigées dans les cases et espaces réservés à cet effet, tout le reste sera ignoré.*

## Exercices

### **Exercice 1. HPF**

#### Question 1.

Soit  $V$  un vecteur de taille  $N$ , c'est à dire indicé de  $0$  à  $N-1$ . On suppose que dans la version de HPF utilisée, les tableaux de scalaires sont automatiquement initialisés à  $0$ , c'est à dire pour ce qui nous concerne que  $V[i]$  vaut zéro après la déclaration de  $V$ .

On considère le code suivant :

```
...  
INTEGER V(N)  
  
V[0]=1  
FORALL (i = 1:N-1)  
    V[i]= V[i-1]*i  
END FORALL  
...
```

#### Question 1.1.

Que vaut  $V[i]$  à la fin de la boucle ?

#### Question 1.2.

Modifier le programme de sorte qu'à la fin de la boucle  $V[i]$  vaille  $i!$ .

Question 2.

Question 2.1. A quelle condition peut on paralléliser la boucle suivante ?

```
...  
DO i=1,n-1  
  X[i]=f(Y[i], Y[i-1])  
END DO
```

...

Question 2.2. Si la condition indiquée à la question 2.1. est satisfaite, de quelles façons peut-on l'indiquer au compilateur HPF ?

Question 3.

Question 3.1. Peut on paralléliser la boucle suivante ?

```
...  
DO  
  X[i]=Y[i-1]  
  Y[i]=Z[i]+K[i];  
END DO
```

...

Question 3.2. Peut on paralléliser la boucle suivante ?

```
...  
FORALL  
  X[i]=Y[i-1]  
  Y[i]=Z[i]+K[i];  
END FORALL
```

....

## Exercice 2. Open MP

On considère le programme OpenMP suivant :

```
...
integer :: a, b, self
a=500
b=0
self=0
!$OMP PARALLEL
    !$OMP PRIVATE(self) FIRSTPRIVATE(a) SHARED(b)
self=OMP_GET_THREAD_NUM();
a=a+self;
b=b+self;
print *, self, ":", a, "/", b
!$OMP END PARALLEL
print *, self, ":", a, "/", b
...
```

Il est exécuté de la façon suivante

```
$ export OMP_NUM_THREADS=2; a.out
```

Question 2.1. Donner un affichage possible de ce programme.

Question 2.2. Quelle est sa traduction à base de *threads* ?

### **Exercice 3. J2EE**

Question 3.1. Expliquer brièvement en quoi consiste une architecture multi-tiers.

Question 3.2. Quels sont les 4-tiers principaux de la plate-forme J2EE ainsi que leurs fonctions (1 ligne maximum par tiers).

### **Problème. Gestion de traces d'exécution via le service Evenements de CORBA**

L'objectif de ce problème est de réaliser un système de gestion et de visualisation de traces d'exécution pour les applications développées avec une bibliothèque de communication inter processus offrant trois primitives : *send*, *receive* et *probe*.

Question 1. Définition du format de traces.

Pour chaque opération, définir la trace portant l'information minimale permettant de construire une représentation graphique de type *time-line* diagram (c'est à dire montrant les communications inter-processus).

`send(int processus_cible, void *buffer, int longueur_du_buffer)`

receive(int processus\_source, void \*buffer, int longueur\_du\_buffer)

int processus\_pret probe(int processus\_a\_tester[], int nb\_de\_processus\_a\_tester)

### Question 2.

Les traces des n processus sont collectées dans n fichiers répartis sur n processeurs.

Si on souhaite visualiser les traces à l'aide d'un outil (de type Paragraph), il faut réaliser la fusion des n fichiers de trace collectés.

Question 2.1. Quel est le problème posé par l'absence d'horloge globale ?

Question 2.2. Que peut on utiliser pour « remplacer » une horloge globale ?

Question 2.3. Expliquer par un schéma.

Question 3. Recollement des traces.

Document associé à cette question

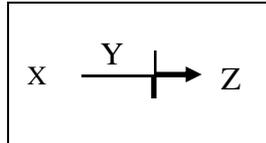
Event Service Specification

October 2004, Version 1.2, formal/04-10-02

OMG, Object Management Group

**NOTE IMPORTANTE POUR LA LECTURE DE CE DOCUMENT**

La notation



signifie que X a une interface de type Y sur Z

Au lieu de regrouper « à la main » les traces dans un fichier global sur la machine pilote, on s'appuie sur le service événements de CORBA pour réaliser cette opération et plus précisément sur le principe d'*Event Channel* décrit au 2.2. du document associé à cette question.

Question 3.1.

Réalisez un schéma montrant les différents processus que vous mettez en place et leurs connexions ainsi que leurs rôles les uns vis à vis des autres.

A large empty rectangular box intended for the student to draw a diagram showing processes and their connections.

Question 3.2.

Donner une version en langage algorithmique libre du programme de visualisation des traces.

A large empty rectangular box intended for the student to write an algorithmic version of the trace visualization program.

Question 3.2.

Donner une version en langage algorithmique libre de la version instrumentée de la fonction *send* qui génère une trace quand on l'utilise.

