Enoncé du partiel ASPD
ESIAL 2, année 2002/2003

Indications : Seuls les documents de cours sont autorisés (cours +document sur PVM). Il faudra justifier toutes vos réponses. À chaque affirmation, il faudra faire un commentaire pertinent.

Un soin tout particulier devra être apporté à la présentation de la copie.

Exercice : Trouver le maximum dans un tableau.

Question 1 : Ce programme calcule le maximun dans un tableau. Compléter le programme :

```
#include <stdio.h>
#include "pvm3.h"
#define NG  15         /* taille du groupe NG+1 */
#define GROUP  "GR"   /* nom du groupe  */
#define ROOT  0       /* index du premier membre du groupe  */
#define COUNT 100     /* taille des sommes partielles */
#define DISTAG  20
#define RESTAG  20
#define ANYTAG -1
int main(){    //  la fonction principale.....
   int max,gid,tids[NG],TG[(NG+1)*COUNT],TL[COUNT],i;
   int info, RES[NG+1];

/* 1) transformer le processus unix en une tache PVM  */

// Si il y a une erreur quitter pvm + sortir du programme
 if (                     ){ printf("erreur ");                         }

 /* 2) Rejoindre le groupe de nom  GROUP  + recuperer mon identifiant du groupe*/
 if ((                            ) <0 )
     {                // si il y a erreur quitter pvm + sortir du programme
                        }
 /* 3) Creation des autres processus si je suis le premier membre du groupe */
 if (              )
     {    // initialisation de facon al{\'e}ratoire du vecteur
         srand(pvm_mytid());
         for (i=0;i<(NG+1)*COUNT; i++) {  TG[i]=(int)(rand()%10000) ; }
         // 3.1) la premiere tache lance les autres taches..
         if (                        ) <NG ) // si il y a erreur quitter pvm + sortir du programme
```

```
                 { printf("erreur ");                              } }

 /*  4) synchronisation du groupe */


 /* 5) diffuser les donnees   */
 if (                    ){ // le premier membre du groupe distribue les donn{\'e}es
     for (i=0; i!=NG;i++)
          { // 5.1)  envoyer les information




          }
        for (i=0; i!=COUNT;i++) TL[i]=TG[i];
 }else { //5.2 reception des donn{\'e}es


        }

 /* 6 ) calculer le Max local  */
 max= TL[0];  for (i=1;i<COUNT;i++) {  if (max < TL[i]) { max = TL[i];}  }
/*  7) Calculer le max des max locals  */
 if (                     ){
   RES[0]=max ; // 7.1 le premier membre du groupe receptionne tous les max locaux
   for (i=0; i!=NG;i++)
        {

                                                 }
   } else    { // 7.2)  envoyer le max locaux



             }

 if (gid==ROOT){ // 7.3) le premier membre du groupe je calcule le max global
     max= RES[0];  for (i=1;i<NG+1;i++)   if (max < RES[i]) { max = RES[i];}
    }
 /* 8) AFFICHAGE du  resultat par le premier membre du groupe*/
 if (                    )   {printf("max =%d  \n ",max,gid); }

 /* 9) quitter le groupe */


 /* 10) quitter pvm */

}
```

Question 2 : Pouvez-vous expliquer quel est le rôle du paramètre `stride` dans la fonction `pvm_pkint`. Donnez un petit exemple. Soyez précis.

Rappel : `int info= pvm_pkint(int *ip, int nitem, int stride)`

Question 3 : Donnez l'algorithme séquentiel qui calcule le maximum dans un tableau à n éléments. Quelle est sa complexité ?

Question 4 : Donnez la complexité de l'algorithme écrit en PVM donné dans la question 1 avec n nombre d'élements du tableau et p le nombre de processeurs (*Comptez simplement les comparaisons*). Calculer le temps parallèle, le facteur d'accélération, l'efficacité.

Question 5 : Peut-on améliorer la rapidité de la partie 7 du programme de la question 1. Si vous répondez par non, expliquez pourquoi. Si vous répondez par oui, decriver le mécanisme puis récrivez cette partie en PVM et enfin donnez à nouveau la complexité de l'algorithme.

Question 6 : Dans la partie 5 du programme de la question 1 la distribution des données se fait en utilisant des fonctions qui gèrent des communications point-à-point. Peut-on remplacer ces fonctions par des fonctions qui qui gèrent des communications de groupes. Si vous répondez par oui, récrivez cette partie en PVM. Si vous répondez par non, expliquez pourquoi.

Question 7 Même question que pour la question mais nous considérons la partie 7 du programme de la question 1.

Exercice : ordonnancement de tâches sur p processeurs identiques.

L'objectif de cet exercice est de déterminer la performance d'un ordonnancement de tâches sur p processeurs. Nous sommes dans la situation suivante :

- Le contrôleur reçoit un ensemble de tâches $T = \{1, 2, \ldots, m\}$. Notons $dispo(i)$ la date de la tâche i à laquelle la tache i peut s'executer (La tâche i ne s'execute pas avant la date $dispo(i)$).
- Aucune dépendance entre les tâches entre elles.
- Chaque tâche i demande exactement $w(i)$ unités de temps pour s'exécuter.

On définit un ordonnancement σ qui attribue à une tâche i une machine $m(i)$ et une date de début d'exécution $t(i)$. L'algorithme est le suivant : le contrôleur trie de facon croissante les tâches en fonction de la fonction *dispo*. En considérant les taches dans l'ordre de cette liste, l'algorithme impose à une tâche i de s'exécuter à l'étape $t(i)$ sur la machine $m(i)$ si $t(i) \geq dispo(i)$ et si la machine $m(i)$ est disponible à l'étape $t(i)$ le plus tôt possible parmi toutes les p machines libres.

Voici un exemple :

tâches	1	2	3	4	5	6	7	8
durée	3	5	2	4	3	3	1	2
$dispo(i)$	4	1	6	1	6	0	8	5

On supposera qu'il y a trois processeurs. Voici la liste triée des tâches = $\{6, 2, 4, 8, 5, 3, 7\}$. La tâche 6 arrive à l'étape 0 sur le processeur P_1. La tâche 2 arrive à l'étape 1 sur le processeur P_2 (car $dispo(2) = 1$ et P_2 était libre).

Question 1 : Continuer l'exemple (ordonnancer toutes les tâches de l'exemple). Pour être plus clair, faire un dessin.

Notons σ l'ordonnancement obtenu par l'algorithme précédent et $D(\sigma) = max\{t(i) + w(i)|i = 1, \ldots, n\}$ la date où l'ordonnancement σ se termine. Notons D_{opt} la durée de l'ordonnancement optimal.

Question 2 : Donner la définition formelle de D_{opt}. En déduire un minorant de $D(\sigma)$.

Question 3 : Donner une borne inférieure de D_{opt} en fonction des durées des tâches.

Question 4 : Donner une borne inférieure de D_{opt} en fonction de la fonction *dispo*.

Question 5 : Soit j une tâche de E se terminant en dernier.

Question 5.1 : Expliquer pourquoi la tâche j n'a pas été exécuté aux dates $dispo(j), \ldots, t(j) - 1$. Que pouvez-vous en déduire sur $(t(j) - dispo(j))$ par rapport aux durées de toutes tâches ?

Question 5.2 : Calculer la valeur β de l'inégalité suivante $D(\sigma) \leq \beta D_{opt}$. Que concluez-vous ?

Parallélisation d'un code.

Voici un code séquentiel (T(i) et S(i) sont les noms des instructions) :

```
Pour i= 1 à N faire
S(i) :    a(i)=b(i)+c(i)
T(i) :    a(i+1)=a(i)+ 2c(i)
Fin de la boucle
```

Question 1 : Exécuter ce programme sur cet exemple avec $N = 4$ et avec

i	1	2	3	4
$a(i)$	9	4	5	2
$b(i)$	3	7	6	1
$c(i)$	3	0	8	5

Question 2 : Paralléliser ce code sachant qu'on dispose de n processeurs. (*Indication : si vous voulez, faites un graphe de dépendances des tâches (cf début du cours sur l'ordonnancement)*).