EISTI CERGY - PAU	Cycle ingénieur 2ème année Examen de Programmation système *Florent* DEVIN, *Mohamed* MAACHAOUI	
	Matière : **Programmation système**	*Date* : **Mai 2016**
	Appareils électroniques et documents interdits	*Durée de l'épreuve* : **3 heures**
	Ordinateur portable et documents électroniques autorisés	*Nombre de pages du sujet* : **2**

1 Préambule

Cet examen n'est pas uniquement un examen de code, mais de réflexion. À ce titre, il vous est demandé de faire une analyse du système avant de vous lancer dans le code. Une analyse bien faite est un pré requis indispensable avant de commencer à coder. Vous devez donc rendre en plus des différents codes demandés, un rapport expliquant clairement votre analyse des différentes parties dans un fichier séparé.

N'oubliez pas de lire le sujet en entier, avant de commencer à répondre aux questions. Certaines questions pouvant vous orienter dans le choix de votre conception.

Par ailleurs, il vous est demandé de produire un code correct. C'est à dire qui compile, et sans warning (en utilisant l'option `-Wall`).

2 Monitoring de calcul

La programmation multi-processus permet d'accélerer les traitements. Dans cet examen, nous nous proposons de résoudre un cas pratique de le programmation multi-processus.

Avec l'avènement de la programmation multi-processus, l'utilisation de programme de *monitoring* est devenu indispensable pour visualiser l'état d'un système. Le moniteur est le point d'entrée de l'application, il permet à l'utilisateur souhaitant visualiser le système de fournir des informations nécessaire à la visualisation. Traditionnellement, ce moniteur possède une interface web, mais dans notre cas des affichages consoles suffiront. Le moniteur collecte les données envoyées par les différents (sous)processus, réalise une synthèse, puis présente les informations à l'utilisateur. Les moniteurs peuvent être, ou non, dédié à un système. Pour plus de simplicité, nous nous concentrerons sur un moniteur dédié à la surveillance de calcul. Le calcul n'étant pas l'objectif de l'examen, nous nous contenterons d'implémenter une "bête" somme d'entier distribuée sur m processus. Toutes les 2 secondes, chaque processus calculateur enverra un rapport. Ce rapport sera constitué de deux parties : la somme partielle calculée ; le temps d'activité du processus. Pour simuler le "dur" calcul, vous mettrez un temps de pause entre chaque addition de 1 seconde.

Le moniteur doit dans un premier temps connaître les processus qu'il doit "monitorer". Une fois l'ensemble des processus répertoriés, il devra faire une synthèse de l'état des processus. Pour des raisons de "performance" cette synthèse n'est effectué que toutes les 3 secondes. Cependant, il doit fournir un état du système à chaque fois que l'utilisateur lui demande. Le rapport que le moniteur doit fournir contient les informations suivantes :

— Nombre de processus calculateurs
— Somme totale partielle calculée
— Pour chaque processus
  — Somme partielle calculée
  — Temps d'exécution

Exercice 1 (Analyse : 5 points ; code : 7 points).

a. Faire l'analyse du moniteur ;

b. faire l'analyse des processus calculateur ;

c. coder le moniteur ;

d. coder les calculateurs ;

e. créez un programme lanceur capable de lancer le moniteur et les m processus de calcul, ainsi que les bornes de départ et de fin des nombres à sommer. Vous veillerez à ce que la répartition entre chaque processus de calcul soit la plus équitable possible.

3 Stratégie de l'échec

Dans un monde totalement distribué, chaque processus est localisé sur des machines différentes. Ces machines pouvant tomber en panne, ou n'être plus accessible, il faut prévoir le cas que les processus puisse tomber en panne. Nous ne vous demandons pas de vous occuper de la partie distribution, mais de vous occuper de mettre en place, la gestion de l'échec.

Exercice 2 (Analyse : 3 points ; code : 1 point).

a. Quels sont les impacts sur le moniteur (justifiez et proposez une solution) ?

b. Quels sont les impacts sur les calculateurs (justifiez et proposez une solution) ?

c. Coder les modifications du moniteur.

4 *Evil monkey*

Pour tester ce type de système, il existe une solution. Il s'agit d'implémenter un processus de type *evil monkey*. C'est à dire un processus qui toutes les x secondes (x étant un nombre aléatoire), tue un processus parmi tous les processus concernés (sauf bien sûr le processus *evil monkey*).

Exercice 3 (Analyse : 2 points ; code : 2 points).

a. Faire l'analyse du processus *evil monkey*.

b. Coder ce processus.