ECU : Programmation système

Enseignant :
Dr Moustapha BIKIENGA

Buts et principes
Les étudiants travailleront en binômes. Cette évaluation s'articule autour de trois exercices.

A rendre
Vous devez créer un fichier unique ZIP, nommé par exemple "Travail_groupe_xx_xx.rar" (xx_xx_xx =concaténation des noms prénoms de l'étudiant). Cette archive doit contenir tout fichier relatif à votre projet. Pour chaque exercice créer un répertoire
L'archive doit contenir les documents (pdf, doc, odt, ppt, …) permettant notamment d'identifier les membres du binôme avec les adresses. L'archive doit également contenir les codes sources commentés, les exécutables, rapports, …

Date limite
La date limite de remise de projet doit impérativement être respectée :

le dimanche 03 février 2004 au plus tard.

Bon travail !

Exercice 1 (6 points)– Fichiers, sémaphores, threads, processus lourds

1-Ecrire un programme qui ouvre un fichier à l'aide de la primitive open. Par la suite le programme crée deux processus légers. Le premier écrira des nombres pairs et le second des nombres impairs.
2-Ecrire un programme qui ouvre un fichier à l'aide de la primitive open. Par la suite le programme crée deux processus légers. Le premier écrira des nombres pairs et le second des nombres impairs. On synchronisera les deux processus légers pour que les nombres soient écrits dans l'ordre croissant.
3- Ecrire un programme qui ouvre un fichier à l'aide de la primitive open. Par la suite le processus (père) crée un autre processus (fils). Le premier (le père) écrira des nombres pairs et le second (le fils) des nombres impairs.
4- Ecrire un programme qui ouvre un fichier à l'aide de la primitive open. Par la suite le processus (père) crée un autre processus (fils). Le premier (le père) écrira des nombres pairs et le second (le fils) des nombres impairs. On synchronisera les deux processus pour que les nombres soient écrits dans l'ordre croissant.

Exercice 2 (7 points)– Concurrence

Le but de cet exercice est de résoudre et simuler un problème de concurrence à savoir le problème du barbier.

Problème du barbier. La boutique du barbier est composée d'une salle d'attente contenant n chaises et du salon où se trouve la chaise du barbier. Lorsque le barbier a fini de raser un client, il fait entrer le client suivant dans le salon. Si la salle d'attente est vide, le barbier s'y installe pour dormir. Si un client trouve le barbier endormi, il le réveille. Si non, il s'installe dans la salle d'attente s'il reste de la place (et rentre chez lui sinon).

1- Ecrivez le code du client et du barbier sans vous préoccuper de synchronisation, mais simplement des opérations que veulent réaliser les processus. En particulier, ignorez pour l'instant que le barbier dort parfois.

2- Il s'agit maintenant d'ajouter les synchronisations nécessaires au programme écrit à la question précédente. Le premier problème `a résoudre est une condition de compétition entre les clients lorsqu'ils rentrent dans la salle d'attente. Corrigez ce problème.

3- Assurez-vous ensuite que le barbier ne commence pas à couper les cheveux tant que le client n'est pas prêt.

4- Enfin, assurez-vous ensuite que le client ne s'assoit pas sur le siège tant que le barbier n'est pas prêt.

5-Proposer une simulation (programme C) de votre solution en utilisant des threads (processus légers).

6-Proposer une simulation (programme C) de votre solution en utilisant des processus lourds.

Exercice 3 (7 points)- Programmation réseau

On considère un fichier qui contient un nombre à 1000 chiffres :

731671765313306249192251196744265747423553491949349698352031277450632623957831801698480186947885184385861560789112949495459501737958331952853208805511125406987471585238630507156932909632952274430435576689664895044524452316173185640309871112172238311362229893423380308135336276614282806444486645238749303589072962904915604407723907138105158593079608667017242712188399879790879227492190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048395864467063244157221553975369781797784617406495514929086256932197846862248239722413756570560574902614079729686524145351004748216637048440319989000889524345065854122758866688116427171479924442928230863465674813919123162824586178664583591245665294765456828489128831426076900422421902267105562632111110937054421750694165896040807198403850962455444362981230987879927244284909188845801561660979191338754992005240636899125607176060588611646710940507754100225698315520005593572972571636269561882670428252483600823257530420752963450

On remarquer par exemple que les quatre chiffres adjacents du nombre à 1 000 chiffres qui ont le plus grand produit sont 9 x 9 x 8 x 9 = 5832

Il faut programmer un client et un serveur.
Au démarrage du serveur il faut indiquer **Y.** Le client se connecte au serveur et demande **X** chiffres. Le serveur lui envoie **X** chiffres extrait du nombre à 1 000 chiffres. Le serveur envoie également **Y**. Le client trouve les **Y** chiffres adjacents qui ont le plus grand produit. Le client renvoie au serveur les **Y** chiffres adjacents qui ont le plus grand produit ainsi que la valeur du produit. Le serveur enregistre les résultats de chaque client.