Examen de programmation système
2ème année ingénieurs EISTI
Sujet 1 – Durée 1 h 30
Novembre 2007

Exercice 1 : question de cours

On dispose d'une fonction void remplir (void) (dont on n'a pas le code source) qui génère un certain volume d'octets sur la sortie standard. On voudrait écrire un programme qui récupère ces octets dans un tableau (on suppose que le volume est inférieur à VERY_LARGE_SIZE octets) pour pouvoir leur appliquer certains traitements ultérieurement. Voici une première proposition pour réaliser cette sorte de redirection :

```
int main()
{
  char buf[VERY_LARGE_SIZE];
  int desc[2], n;
   pipe(desc);
   dup2(desc[1], 1);
   close(desc[1]);
   mystere();
   close(1);
   n = read(des[0], buf, VERY_LARGE_SIZE);
...
}
```

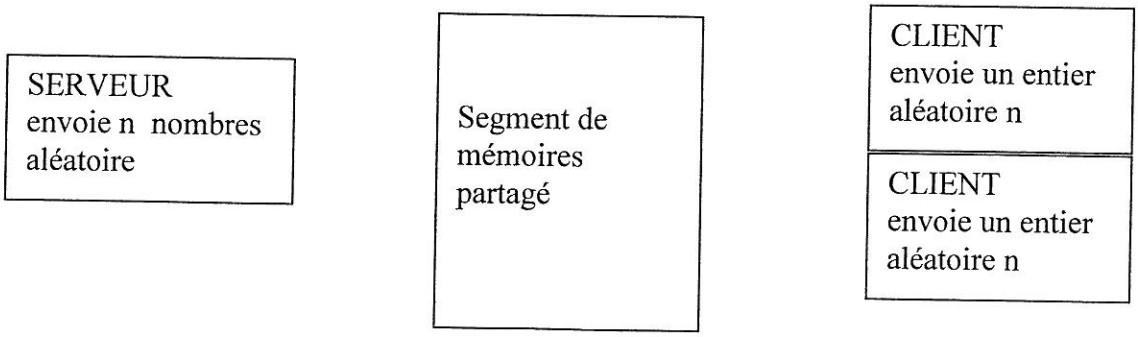
1) Le programme proposé fonctionne lorsque le volume d'octets générés par la fonction remplir() est petit, mais ne se termine pas lorsque le volume devient trop important (tout en restant inférieur à VERY_LARGE_SIZE).

Expliquez pourquoi on observe ce comportement. À quoi correspond la valeur seuil à partir de laquelle le programme ne fonctionne plus ?

2) Proposez une nouvelle version du programme qui fonctionne pour tout volume d'octets compris entre 0 et VERY_BIG_SIZE. Note : on pourra utiliser un processus supplémentaire au besoin.

Exercice 2 : conception

On considère l'architecture logicielle suivante : une application SERVEUR et 2 applications CLIENT indépendants.



Le serveur attend des questions de la part des clients. Un client demande au serveur de lui envoyer n nombres aléatoires compris entre 1 et n. La valeur n est généré aléatoirement par le client, elle est comprise entre 1 et NMAX

1) Faire l'analyse de ce système et donner toutes les structures de données et les primitives à utiliser en justifiant leurs paramètres

2) Si on remplace le segment de mémoire partagé par l'utilisation des tubes ordinaires, indiquez les changements à faire dans l'architecture.

Exercice 3 : implémentation

```
/**
* CACHE D'OBJETS EN MEMOIRE PARTAGEE
* On réalise un cache d'objets en mémoire, ce cache étant partagé
* par plusieurs processus. On NE s'intéresse PAS dans l'exercice
* à la façon dont on arbitre l'accès à ce cache.
*
* Le cache est sensé effectuer un rapprochement de données initialement
* situées dans une base de données lointaine. Pour simuler cette base de
données
* on utilise un tableau statique d'objets.
* On doit pouvoir cependant implémenter les fonctions de base du cache et
vérifier
* leur fonctionnement.
*
* Le cache est piloté par une API décrite par deux fonctions :
* - getObject : récupère un objet duc ache ou de la base par défaut.
* - clearObject : nettoie le cache pour un objet particulier
*
* On vérifie le fonctionnement par un ensemble de fils qui iront chacun
chercher
* un objet dans une position indexée selon leur numéro dans plusieurs
situations :
* - premier accès au cache
* - deuxième accès au cache
* - nettoyage du cache
* - accès après nettoyage
*/
```

Pour réaliser l'exercice, vous devez avoir récupéré la carcasse de source diffusée sur l'activité Devoir de Programmation Système correspondant à votre numéro de sujet.

Réalisez, programmez et complétez la carcasse pour obtenir le fonctionnement souhaité par l'application.

A la fin de l'examen, vous disposerez **d'un quart d'heure** pour :

- Transférer l'archive ZIP sur deathstar (sur votre compte).
- Exécuter la commande `examprogsy <monarchive>` sous votre compte pour transmettre votre devoir au professeur (VF).