Examen de juin

Durée : 3 h (informatique) et 2 h (EEA) — O. Dalle et R. Rousseau
Documents : seulement les copies des supports de cours.
Aucun livre ou corrigé de TPs ou TDs — 3 juin 2005

Commencez par remplir le cadre de la copie d'examen avec votre nom, prénom et cachetez le. L'épreuve comporte deux parties. La première partie porte sur le cours et les TPs. La seconde partie porte sur le cours et les TDs. Les étudiants du Master EEA ne doivent répondre qu'aux questions qui les concernent et indiquer sur leur copie « EEA ». Ils doivent quitter la salle au bout de deux heures. Les étudiants d'informatique doivent répondre à toutes les questions en trois heures. Le barème est indiqué pour chaque question en minutes et points, sachant que les étudiants d'informatique sont notés sur 30 points et ceux d'EEA sur 20 points. Le traitement des erreurs pourra utiliser les macros vues en TP.

PREMIERE PARTIE

Question 1 (Informatique et EEA) *(3 points ≃ 18 mn)*

On considère un système client-serveur avec des sockets TCP en mode connecté.

Compléter sur la copie la fonction `main` du serveur aux points marqués */* N.??? */*. (On n'écrira que les instructions qui manquent aux points N).

```
#include "params.h"
void clientProcessing(int clisock, struct sockaddr_in*  cliaddr);
void traitantUsr1(int sig);
int main(void){
  int mess_sock;
  struct sockaddr_in seraddr;
  if((mess_sock= /* 1.??? */ )<0) ERRNO_EXIT;
  seraddr.sin_family = /* 2.??? */;
  seraddr.sin_addr.s_addr = /* 3.??? */;
  seraddr.sin_port = htons(PORT_SERVEUR_UDP) /* défini dans params.h */ ;
  /* Attachement d'un nom à la socket mess_sock 4.??? */
  /* Transformation de la socket en socket de service 5.??? */
  while(1){
    int clisock;
    struct sockaddr_in cliaddr;
    int cliaddr_len = sizeof(cliaddr);
    /* connexion d'un client 6.??? */
    int child_pid;
    if(!(child_pid= fork())){
      /* fermeture de la socket inutilisée 7.??? */
      clientProcessing(clisock, &cliaddr);
      exit(0);
    } else if (child_pid>0) {
      /* fermeture de la socket inutilisée 8.??? */
      /* association du traitantUsr1 au signal SIGUSR1 9.???*/
    } else ERRNO_EXIT;
  }
  exit(0);
}
```

Question 2 (Informatique et EEA) *(6 points ≃ 36 mn)*

La procédure `clientProcessing` réalise le dialogue avec un client connecté par un sous-processus SPS du serveur. Le protocole est le suivant :

- si SPS reçoit la chaîne `"U"`, il envoie à son père le signal `SIGUSR1` ;
- si SPS reçoit la chaîne `"D"`, il affiche le texte qui suit sur la voie standard de sortie. Pour lire le texte sur la socket, SPS lit d'abord sa taille en octets indiquée par 10 caractères représentant un nombre décimal cadré à droite et précédé d'éventuels espaces. La taille est immédiatement suivie des octets du texte.

- si SPS parvient à réaliser l'une des tâches précédente demandée par le client, il lui renvoit un compte-rendu d'un caractère 'S', sinon un compte-rendu 'F' d'échec.
- si SPS reçoit la chaîne `"E"`, il termine la communication avec le client sans compte-rendu.
- si le compte-rendu est 'F', cela termine aussi la communication avec le client.

Ecrire le code **commenté** de la procédure `clientProcessing` .

Question 3 (Informatique et EEA) *(3 points ≃ 18 mn)*

Question 3.1 *(1 point ≃ 6 mn)*

Plusieurs processus SPS de la question précédente peuvent servir plusieurs clients en parallèle. On décide maintenant que le traitant du signal SIGUSR1 par le père provoque la terminaison du serveur, c'est-à-dire de tous ses processus. Expliquer en français comment réaliser cette terminaison, pour que tous les processus du serveur se terminent proprement.

Question 3.2 *(2 points ≃ 12 mn)*

L'écriture simultanée sur la voie standard de sortie par les processus SPS pose un problème. Lequel ? Programmer une solution à ce problème avec des IPC système V.

Question 4 (Informatique seulement) *(6 points ≃ 36 mn)*

On considère les déclarations suivantes :

```
typedef void FILE_HANDLER ( const char *pathName, const struct stat *inode );
void findExec (char *pathName, FILE_HANDLER *func);
/* Applique la procédure func au fichier pathName. Si le fichier est un répertoire,*/
/* applique récursivement func à tous les fichiers de ce répertoire (sauf "." et ".."). */
/* On suppose que pathName repère un emplacement mémoire assez grand pour contenir*/
/* le nom de chemin le plus long rencontré lors du parcours récursif. */
/* En cas de lien symbolique, la fonction func est appliquée au lien lui-même.*/
```

Ecrire le code **commenté** de la procédure `findExec`.

DEUXIEME PARTIE

Question 5 (Informatique et EEA) *(4 points ≃ 24 mn)*

On souhaite donner une solution au problème du producteur/consommateur à l'aide d'un moniteur. Votre moniteur devra comporter :

- une partie commune pour les déclarations de variables (et leur initialisation si besoin) ;
- une procédure `ajouter_obj(objet)` pour l'insertion d'un objet (par un producteur) dans le tampon circulaire ;
- et une fonction `retirer_obj()` retournant (au consommateur) le premier objet consommable depuis le tampon circulaire.

Le tampon circulaire est une variable tableau de taille N nommée `buffer`.

Votre moniteur pourra s'appuyer sur deux routines prédéfinies `add(buffer,objet)` et `remove(buffer)` qui prennent en charge respectivement l'ajout d'un objet à la fin du tampon et le retrait d'un objet au début du tampon, mais SANS vérifier si le tampon est plein ou vide (à vous donc d'écrire ces vérifications dans les routines du moniteur).

Question 6 (Informatique et EEA) *(4 points ≃ 24 mn)*

Question 6.1 *(0,5 points ≃ 3 mn)*

Soit l'adresse IP suivante d'une machine appelée "une.machine.fr" au format «point-décimal» : 138.96.232.4.

On note $O_1O_2O_3O_4$ l'entier sur 32 bits formé des quatre octets O_1, O_2, O_3 et O_4, avec O_1 l'octet de poids le plus fort et O_4 l'octet de poids le plus faible.

Quelles sont les valeurs des octets O_1, O_2, O_3 et O_4 lorsqu'ils contiennent l'entier binaire sur 32 bits au format réseau correspondant à l'adresse IP de "une.machine.fr" ?

Question 6.2 *(0,5 points ≃ 3 mn)*

Une machine est équipée d'un processeur qui stocke en mémoire les 4 octets d'un entier sur 32 bits de la façon suivante, lorsque l'on regarde les quatre octets consécutivement dans l'ordre décroissant des adresses physiques : $O_3O_1O_4O_2$.

Supposons qu'un message UDP en provenance de la machine "une.machine.fr" de la question précédente soit reçu par une machine "autre.machine.fr" ayant ce type processeur, à l'aide de l'instruction suivante, où `from` est une variable de type `struct sockaddr_in` :

```
nbrecv = recvfrom(sockfd, buff, buflen, 0, (struct sockaddr *) &from, &fromlen);
```

A la suite de cette instruction, quelles seront, dans l'ordre décroissant des adresses pysiques, les valeurs des 4 octets du champ `from.sin_addr.s_addr` ?

Question 6.3 *(1 point ≃ 6 mn)*

Toujours sur la machine "autre.machine.fr", supposons maintenant qu'à la suite de l'instruction `recvfrom` décrite à la question précédente, on trouve les instructions suivantes :

```
from.sin_addr.s_addr++;
printf("En ajoutant 1 j'obtiens :  %s\n",inet_ntoa(from.sin_addr));
```

Quel sera l'affichage produit et pourquoi ? (On rappelle que `inet_ntoa` suppose que son argument est fourni au format réseau).

Question 6.4 *(2 points ≃ 12 mn)*

En supposant que sur la machine "autre.machine.fr", les octets d'un tableau d'octets sont stockés consécutivement dans l'ordre décroissant des adresses physiques de la mémoire (en partant de l'indice 0), donnez le code de la fonction de conversion `ntohl` sur cette machine. Pour éviter les calculs binaires, vous pourrez supposer que les arguments et valeurs de retour de cette fonction sont des tableaux d'octets contenant chacun les 4 octets de l'entier sur 32 bits à convertir ou retourner. La fonction demandée a donc la signature suivante :

```
unsigned char *ntohl(unsigned char *val)
```

Question 7 (Informatique seulement) *(4 points ≃ 24 mn)*

Soit la séquence suivante des accès à la mémoire d'un programme :

`10, 50, 100, 170, 73, 309, 185, 245, 246, 434, 458, 364.`

Question 7.1 *(0,5 points ≃ 3 mn)*

Donnez la séquence des numéros des pages utilisées par le processus en considérant que la taille des pages est de 100 mots.

Question 7.2 *(1 point ≃ 6 mn)*

Calculez le taux de défauts de page pour cette séquence, en supposant que le système a affecté 200 mots de mémoire centrale à ce programme et qu'il utilise un algorithme de remplacement des pages optimal. Vous expliciterez ce calcul en donnant, pour chacun des cadres de mémoire physique affecté au processus et pour chacun des accès à la mémoire, les numéros des pages présentes dans ces cadres à l'instant qui précède juste l'accès à la mémoire.

Question 7.3 *(1 point ≃ 6 mn)*

Quel serait le taux de défauts de page si l'algorithme de remplacement était LRU (Least Recently Used) ? Explicitez le calcul de la même façon que précédemment.

Question 7.4 *(0,5 points ≃ 4 mn)*

Les questions précédentes ont du vous permettre de constater une nouvelle fois que OPT est meilleur que LRU. Mais en pratique on ne l'implémente jamais. Pourquoi ? Et sachant qu'on ne l'implémente jamais, pourquoi s'y intéresse-t-on aussi souvent ?

Question 7.5 *(1 point ≃ 5 mn)*

A l'opposé de OPT, proposez une définition pour l'algorithme PIRE, qui provoque systématiquement le plus grand nombre de défaut de pages ?