Contrôle Systèmes d'exploitation, Réseaux

Jeudi 29 Juin 2017
14h - 17h
Aucun document n'est autorisé

Exercice 1 (15 = 1 + 2 + 3 + 2 + 3 + 1 + 3 pts)

On suppose qu'il existe une fonction `char *classifieur(char *s)` *qui prend comme arguments une suite d'octets (terminée par '\0') et qui retourne un mot (une chaîne de caractères terminée par '\0') déterminant la classe du contenu (p.e. si la suite d'octets est une image de fleur, la fonction retourne "FLEUR").*

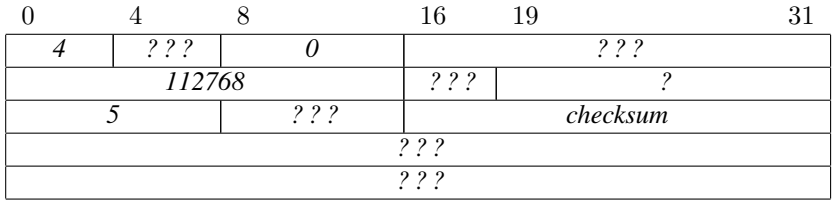
1. *Ecrire une fonction* `char **classement(char **s, int taille)` *(sans thread, ni fork) qui prend en argument un tableau de suites d'octets (*`taille` *est la taille du tableau) et renvoie le tableau des résultats obtenus par la fonction* `classifieur` *appliquée à chaque élément du tableau de suites d'octets.*
2. *Rappeler ce qu'est un système de parallélisme maximal, un système déterminé. On considère que, dans la fonction* `classement` *précédente, chaque appel à la fonction* `classifieur` *est une tâche. Le système de tâches défini par la fonction* `classement` *tel que programmé à la question précédente est-il déterminé ? Si oui, donner le système de parallélisme maximal équivalent. Dessiner son graphe de flot, son graphe de précédence.*
3. *Modifier la fonction* `classement` *pour que chaque appel à la fonction* `classifieur` *soit faite par un thread (on pourra définir une structure pour le passage des arguments).*
4. *On suppose connaître le nombre de cœurs de la machine que l'on indiquera comme une constante du programme. Modifier la fonction précédente pour que, à chaque instant, le nombre de threads en exécution ne dépasse pas le nombre de cœurs de la machine.*
5. *Réécrire la fonction* `classement` *de la première question en créant des processus au lieu de threads (et en utilisant des tubes pour la communication entre processus).*
6. *On veut maintenant que cette fonctionnalité soit accessible à travers un programme serveur. Quelles différences cela fait-il de choisir un protocole TCP ou UDP ? Que choisir ici ?*
7. *Ecrire le code du serveur avec le protocole que vous avez choisi à la question précédente. Après s'être connecté, un client envoie en séquence :*
   — *le nombre de suite d'octets qu'il souhaite classer*
   — *pour chaque suite d'octets, la taille de la suite d'octets puis la suite d'octets*
   Le serveur appelle alors la fonction `classement` *puis retourne au client le tableau des résultats de classement.*

Exercice 2 (2 = 4*0,5)

Une entreprise s'est vu affecter l'adresse IP 99.0.0.0 pour une gestion plus fine de ses sous réseaux, le responsable informatique désire pouvoir affecter une adresse IP propre à chaque sous réseau des 30 succursales.

1. *De quelle classe s'agit-il ?*
2. *Donner et expliquer la valeur du masque de sous réseau correspondant à ce besoin.*
3. *Combien de machines chaque sous réseau pourra-t-il comporter et pourquoi ?*
4. *Quelle est l'adresse de diffusion (broadcast) du sous réseau 12 ?*

Exercice 3 (3) *Une station X souhaite transmettre à une autre station Y un datagramme TCP dont la taille (incluant l'entête TCP) est égale à 3500 octets. La machine X est dans un réseau A alors que Y est dans un réseau C. L'adresse de X est 88.34.72.203. On suppose que la machine Y a comme numéro IP 161.34.201.164. La MTU du réseau A est égale à 3000 octets. Le datagramme envoyé par X quitte le réseau A en passant par un routeur R1, il atteint le réseau B de MTU = 2048 octets. Il passe ensuite par un routeur R2 pour atteindre le réseau C, dont la MTU est égale à 1500 octets. La structure de l'en-tête IP du datagramme dans le réseau A est présentée ci-dessous :*



Déterminez les paquets IP (en précisant l'en-tête de chaque paquet) qui passent dans les réseaux A, B et C. (La somme de contrôle de l'en-tête n'est pas à calculer)

ANNEXE : Fonctions et structures utiles (on ne se préoccupera pas des en-têtes nécessaires

```
int pipe(int pipefd[2]);
pid_t fork(void);
pid_t wait(int *status);

int pthread_create(pthread_t *thread,
        const pthread_attr_t *attr, void *(*start_routine)(void*), void *arg);
int pthread_join(pthread_t thread, void **value_ptr);
void pthread_exit(void *value_ptr);

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);

int sem_init(sem_t *sem, int pshared, unsigned int value);
int sem_wait(sem_t *sem);
int sem_post(sem_t *sem);
int sem_getvalue(sem_t *sem, int *valp);
int sem_destroy(sem_t *sem);

int open(const char *pathname, int flags);
int close(int fd);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int send(int sock, const void *msg, size_t count, int flags);
int recv(int sock, void *msg, int count, unsigned int flags);
int sendto(int sock, const void *msg, size_t count,
      int flags, const struct sockaddr *destinataire,socklen_t adrlen);
int recvfrom(int sock, void *msg, int count,
      unsigned int flags,struct sockaddr *emetteur,socklen_t *adrlen);

int socket(int domaine, int type, int protocole);
int bind(int sock, struct sockaddr * adr, socklen_t size);
int listen(int sock, int backlog);
int accept(int sock, struct sockaddr *adresse, socklen_t *longueur);
int connect(int sock, struct sockaddr *addr serv, socklen_t longueur);

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);

int inet_aton(const char *cp, struct in_addr *in);
char * inet_ntoa(struct in_addr in);
int getsockname(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
struct hostent * gethostbyname(char * hostname);
long gethostid(void);

struct sockaddr {
   short sa_family ;
   char sa_data[14] ;
} ;

struct sockaddr_in {
   short sin_family ;
   u_short sin_port ;
   struct in_addr sin_addr ;
   char sin_zero[8] ;
} ;

struct hostent {
   char * h_name ;
   char **h_aliases ;
   int h_addrtype ;
   int h_length ;
   char ** h_addr_list ;
} ;
```