Institut Galilée — Systèmes d'exploitation, Réseaux
Année 2015-2016 — Licence 2

Contrôle Systèmes d'exploitation, Réseaux

Mercredi 29 Juin 2016 — 14h - 17h
Aucun document n'est autorisé

Exercice 1 (13 = 2 + 3 + 2 + 3 + 3 pts)

On suppose qu'il existe une fonction `char *cat(char *s1, char *s2)` *qui prend comme arguments deux chaînes de caractères (chacune terminée par '\0') et qui retourne la concaténation de ces chaînes.*

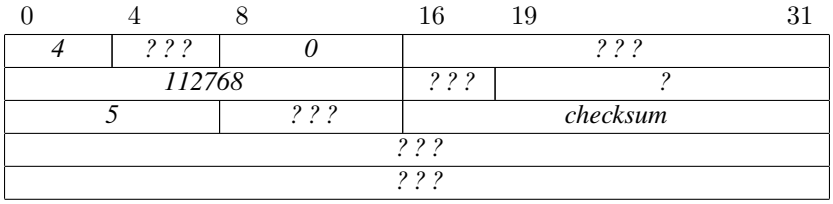
1. *Ecrire un programme (sans thread, ni fork) qui prend en argument un nombre quelconque de noms de fichiers et écrit sur la sortie standard la concaténation du contenu de tous les fichiers.*
2. *Afin d'utiliser au mieux une machine multi-coeurs, on veut écrire un programme de concaténation qui utilise une méthode dichotomique. Le programme doit faire appel à une fonction récursive dont la signature est* `char *concatener(char** ns, int n)` *qui prend comme arguments un tableau de n chaînes de caractères (chacune terminée par '\0'). Cette fonction* `concatener` *a alors le principe suivant :*
   — *si $n = 1$,* `concatener` *renvoie simplement la chaîne passée en argument,*
   — *si $n = 2$,* `concatener` *fait appel à la fonction* `cat` *et retourne le résultat,*
   — *sinon* `concatener` *lance un thread, qui devra concaténer les $\lfloor n/2 \rfloor$ premières chaînes, et un autre thread qui concatènera les autres chaînes, puis retourne la concaténation des deux résultats obtenus.*
   Ecrire un tel programme. Attention à la gestion de la mémoire commune.
3. *On suppose connaître le nombre de coeurs de la machine que l'on indiquera comme une constante du programme. Modifier le programme précédent pour que, à chaque instant, le nombre de threads en exécution ne dépasse pas le nombre de coeurs de la machine.*
4. *Réécrire la fonction* `concatener` *de la deuxième question en créant des processus au lieu de threads (et en utilisant des tubes pour la communication entre processus).*
5. *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 ?*
6. *Ecrire le code du serveur : Après s'être connecté, un client envoie le nombre de fichiers qu'il souhaite concaténer puis les noms de fichiers, le serveur appelle la fonction concaténer puis retourne au client la chaîne de caractères résultant des concaténations.*

Note : $\lfloor n/2 \rfloor$ est la valeur entière par défaut de $n/2$.

Exercice 2 (4 = 2 + 2 pts)

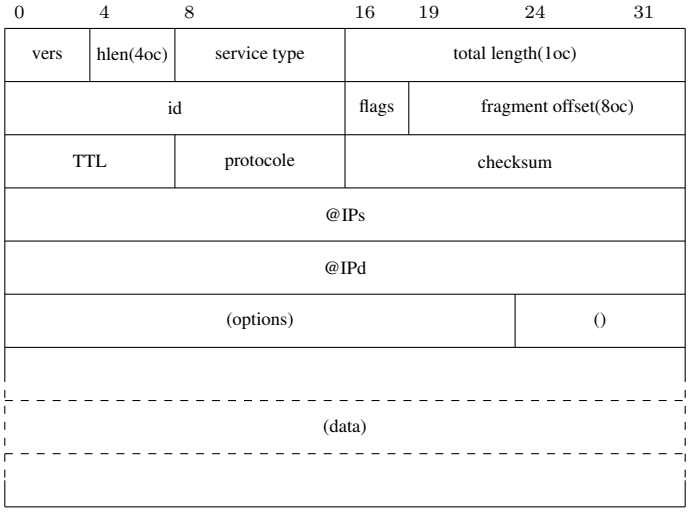
1. *Donnez dans chacun des cas suivants l'adresse du réseau, du sous réseau, le numéro de la machine pour les configurations suivantes :*
   (a) *Adresse IP = 131.27.31.6 ; masque = 255.255.255.0*
   (b) *Adresse IP = 78.15.69.69 ; masque = 255.128.0.0*
2. *Une entreprise s'est vu affecter l'adresse IP 138.27.0.0 pour une gestion plus fine de ses sous réseaux, le responsable informatique désire pouvoir affecter une adresse IP propre à chaque sous réseau des 12 succursales.*
   (a) *De quelle classe s'agit-il ?*
   (b) *Donner et expliquer la valeur du masque de sous réseau correspondant à ce besoin.*
   (c) *Combien de machines chaque sous réseau pourra-t-il comporter et pourquoi ?*

Exercice 3 (3 pts) *Une station X souhaite transmettre à une autre station Y un datagramme TCP dont la taille (incluant l'entête TCP) est égale à 2000 octets. La machine X est dans un réseau A alors que Y est dans un réseau C. L'adresse de X est 121.44.11.103, celle de Y est 83.100.33.22. La MTU du réseau A est égale à 1500 octets. Le datagramme envoyé par X quitte le réseau A en passant par un routeur R1, il atteint le réseau B de MTU = 512 octets. Il passe ensuite par un routeur R2 pour atteindre le réseau C, dont la MTU est égale à 3000 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 1) Structure d'un paquet IPv4 :



(Annexe 2) Fonctions et structures utiles :

```
pid_t fork(void);
int pipe(int tableau[2]);

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_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 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 getchar(void);

int socket(int domaine, int type, int protocole);
int bind(int sock, struct sockaddr *adresse, 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) ;
struct hostent * gethostbyname(char * hostname) ;

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 ;
} ;
```

(Annexe 3) serveur.c :

```c
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <errno.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <strings.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 7777

void init_serveur(void);
void operation(int);

int main(int argc, char *argv[]) {
   init_serveur();
   return 0;
}

void init_serveur(void) {
   int sock, nsock;
   struct sockaddr_in serveur;
   struct sockaddr_in client;
   socklen_t size;

   sock=socket( ???? );
   bzero(&serveur, sizeof(serveur));
   serveur.sin_family = ???? ;
   serveur.sin_addr.s_addr = INADDR_ANY;
   serveur.sin_port = ???? ;

   bind( ???? );
   listen( ???? , 5);

   size = sizeof(client);
   nsock = accept( ???? );
   printf("Connexion du client d'adresse %s et de port %d\n", ???? , ???? );

   operation(nsock);
   close(nsock);

   return;
}

void operation(int sock) {
   ????
}
```