

Contrôle Systèmes d'exploitation, Réseaux

Vendredi 24 Mai 2013
14h - 17h
Aucun document n'est autorisé

Exercice 1 : (5 = 1 + 2 + 1 + 1)

1. Citez les 4 couches du modèle TCP/IP. Nous ne vous demandons pas d'expliquer leur fonctionnalité.
Couches accès réseau, réseau IP, transport, application
2. Quels sont les 2 principaux protocoles de transport dans le modèle TCP/IP ? Indiquez en quelques lignes leurs différences principales, les cas d'utilisation.
TCP et UDP. Voir slides 13 et suivants
3. Une trame Ethernet se compose de 5 parties dans le modèle TCP/IP. Rappelez ce que sont ces 5 parties.
Voir slide 3
4. Qu'est-ce que le MTU ? Que définit-il ?
voir slide 12

Exercice 2 : (5 = 1 + 1 + 1 + 1 + 1)

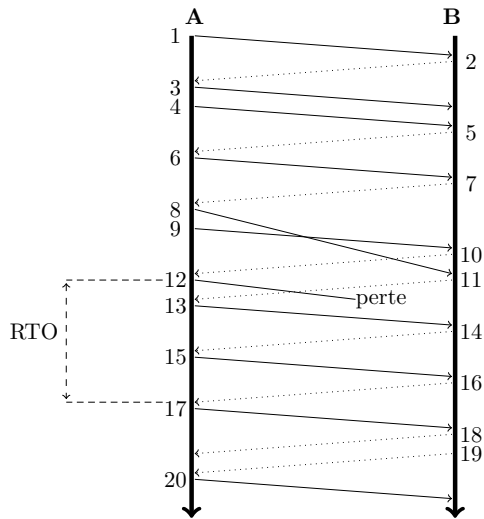
Une entreprise composée de 12 départements se voit affecter l'adresse IP 202.12.123.0. L'administrateur souhaite affecter un sous-réseau à chaque département.

1. De quelle classe d'adressage s'agit-il? Combien de machines cela permet-il d'adresser?
202.12.123.0 donne en binaire : 1100 1010.0000 1100.0111 1011.0000 0000. Donc c'est une classe C (car commence par 110). Le nombre de machines adressables est de 254.
2. En supposant que le nombre de départements de l'entreprise ne va pas tellement évoluer, quel est le masque de sous-réseau optimal ?
12 départements donc un masque de sous-réseau sur 4 bits.
3. Combien de départements peuvent être ajoutés avec ce masque ?
16 - 12 - 2 = 2 puisque 0000 sert au masque de réseau et 1111 pour le broadcast (et déjà 12 départements).
4. Combien de machines chaque département peut-il comporter ?
Il reste 4 bits sur la partie host id donc 16 - 2 = 14 (le 0000 est pour le réseau, 1111 pour le broadcast)
5. Quelle est l'adresse de broadcast du premier sous-réseau
202.12.123.31 = 202.12.123.0001 1111

Exercice 3 : (5)

Nous considérons une connexion TCP entre deux stations A et B. Seul A veut envoyer des données vers B. B n'a rien à envoyer, à l'exception des messages d'acquittement. Pour le transfert de A vers B, la taille des données d'un segment est fixée à 100 octets.

La transmission démarre, nous représentons dans la figure 1 le chronogramme obtenu : complétez les paramètres associés à chaque segment (Numéros de séquence et d'acquittement, flags et Taille des données) dans le tableau fourni en annexe A.



Exercice 4 : Client-Serveur(5 = 1+1+1+1+1)

On considère pour toute cette épreuve les deux programmes B et C dont les codes sont donnés en annexe.

1. Préciser le programme qui correspond au serveur en justifiant le rôle de chacune des primitives associées (écrites soulignées)

B = serveur

voir cours pour la signification des primitives.

2. Même question pour le client.

C = client

voir cours pour la signification des primitives.

3. Le programme qui correspond au serveur est exécuté sur la machine TERRE et le client sur la machine LUNE.

- Dans quel ordre doit-on lancer l'exécution de ces programmes pour avoir une exécution correcte ? Décrivez graphiquement le processus d'exécution.

serveur d'abord. Voir cours pour le processus d'exécution

- Quels doivent être les paramètres du programme client pour que le serveur affiche :

Message reçu : Test de transfert

Message sans espace : Testdetransfert

terre 50000 "Test de transfert"

4. Compléter les codes (en rappelant les numéros de ligne).

voir codes ci-dessous

5. On substitue dans le code du programme B les lignes 31 à 40 par les lignes suivantes :

```

31 listen(s, 3);
32
33 i = sizeof(struct sockaddr_in);
34 for (;;) { /* Boucle infinie */
35     t = accept(s, (struct sockaddr *) &isa, &i);
36     if (t < 0) {
37         perror("accept"); exit(1);
38     }
39     switch(fork()) {
40         case -1 :    perror("fork"); close(s); close(t); exit(1);
41         case 0 :    close(s); travaille(t); close(t); exit(0);
42         default :   close(t);
43     }
44 }
```

Expliquer pourquoi on utilise la fonction `travaille(t)` avec le paramètre `t` et non le paramètre `s` à la nouvelle ligne 41 ?

La variable `s` correspond au socket principal, fermer `s` revient à fermer l'écoute globale, fermer la socket `t` revient à fermer la connexion avec le client en cours de traitement.

Fonctions et structures utiles:

```
1 int socket(int domain, int type, int protocol);
2 int connect(int sockfd, struct sockaddr *serv_addr, socklen_t addlen);
3 int bind(int sockfd, struct sockaddr *my_addr, socklen_t addlen);
4 int listen(int sockfd, int size);
5 int accept(int sockfd, struct sockaddr *addr, socklen_t addlen);
6 struct hostent *gethostbyname(const char *name);
7
8 struct hostent {
9     char *h_name;           /* official name of host */
10    char **h_aliases;       /* alias list */
11    int h_addrtype;         /* host address type */
12    int h_length;           /* length of address */
13    char **h_addr_list;     /* list of addresses */
14 };
15 #define h_addr h_addr_list[0] /* for backward compatibility */
16
17 struct sockaddr {
18     sa_family_t sa_family;
19     char sa_data[14];
20 };
21 struct sockaddr_in {
22     short sin_family;
23     u_short sin_port;
24     struct in_addr sin_addr;
25     char sin_zero[8];
26 };
```

Annexe B

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/utsname.h>
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <netinet/in.h>
8 #include <arpa/inet.h>
9 #include <netdb.h>
10 #include <unistd.h>
11
12 #define PORINUM 50000
13 void travaille (int s) ;
14
15 int main(int argc, char *argv[]) {
16     int s, t, i;
17     struct sockaddr_in sa, isa;
18
19     if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
20         perror("socket"); exit(2) ;
21     }
22
23     sa.sin_family = AF_INET;
24     sa.sin_port = htons(PORINUM);
25     sa.sin_addr.s_addr = INADDR_ANY;
26
27     if (bind(s, (struct sockaddr *)&sa, sizeof(struct sockaddr_in)) < 0) {
28         close(s); perror("bind") ; exit(3) ;
29     }
30
31     listen(s, 1);
32
33     i = sizeof(struct sockaddr_in);
34     for (;;) { /* Boucle infinie */
35         t = accept(s, (struct sockaddr *) &isa, &i);
36         if (t < 0) {
37             perror("accept"); exit(1);
38         }
39         travaille(t);
40     }
41 }
42
43 void travaille (int s) {
44     char msg[256] ;
45     char new_msg[256] ;
46     char *dst, *src;
47     int n;
48
49     n = read(s, msg, 100) ; /* On ne lit que les 255 premiers caracteres */
50     msg[n] = '\0';
51     printf("Message_recu_: %s\n", msg) ;
52
53     /* Portion de code pour garder la chaine d'entree sans espace */
54     /* isblank(c) retourne vrai si le caractere c est un espace ou une tabulation, faux sinon */
55     src=msg ;
56     dst=new_msg ;
57     while (*src != '\0') {if (!isblank(*src)) {*dst=*src; dst++;}; src++;};
58     *dst='\0';
59
60     /* Affichage du message sans espace */
61     printf("Message_sans_espace_: %s\n", new_msg) ;
62     close(s) ;
63 }
```

Annexe C

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/utsname.h>
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <netinet/in.h>
8 #include <arpa/inet.h>
9 #include <netdb.h>
10 #include <unistd.h>
11
12 #define PORTNUM 0
13
14 int main(int argc, char *argv[]) {
15     struct sockaddr_in sa;
16     struct hostent *hp;
17     int a,s,t ;
18
19     if (argc != 4) {
20         perror("Il faut 3 parametres\n") ; exit(2) ;
21     }
22
23     if ((hp = gethostbyname(argv[1])) == NULL) {
24         perror("gethostbyname") ; exit(1);
25     }
26
27     if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
28         perror("socket") ; exit(3);
29     }
30
31     sa.sin_family = AF_INET;
32     sa.sin_port = htons((u_short)atoi(argv[2]));
33     memcpy(&sa.sin_addr.s_addr, hp->h_addr, hp->h_length);
34
35     if (connect(s,(struct sockaddr *) &sa, sizeof sa) < 0) {
36         close(s); perror("connect"); exit(4);
37     }
38
39     if (write(s, argv[3], strlen(argv[3])) != strlen(argv[3])) {
40         perror("Write on socket") ; exit(3) ;
41     }
42
43     close(s) ;
44 }
```