

**NOM Prénom :**  
**Nunéro Étudiant :**

Université Paris Cité

Année 2022–2023

## **Programmation Réseaux - Partiel**

Durée : 1h30  
Aucun document n'est autorisé

Lorsque vous aurez à modifier du code, vous pourrez juste écrire les lignes à modifier en précisant leur numéro ou les lignes à ajouter en précisant entre quelles lignes elles s'insèrent.

1. Donnez une commande Unix pour connaître le nom de la machine sur laquelle on se trouve :

**Réponse :**

2. Donnez une commande pour connaître les adresses IP de la machine `www.ietf.org` :

**Réponse :**

3. Quel enchaînement de commandes Unix utiliser pour déterminer le numéro de port correspondant au service `ftp` ?

**Réponse :**

4. Quel service rend un serveur DNS ?

**Réponse :**

5. Sur combien d'octets sont codées les adresses IPv6 ?

**Réponse :**

6. La commande `ip` a sur lulu donne le résultat suivant :

```
micheli@lulu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 4a:49:43:49:79:bf brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.70.236/24 brd 192.168.70.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fdc7:9dd5:2c66:be86:4849:43ff:fe49:79bf/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4849:43ff:fe49:79bf/64 scope link
        valid_lft forever preferred_lft forever
```

Donnez une adresse IPv4, une adresse IPv6 et l'adresse MAC de l'hôte lulu :

**Réponse :**

7. Quelle commande lancer depuis le terminal pour se connecter en IPv4 à un service sur le port 2222 actif sur la machine `lampe` sous un système Unix (au sein des machines de l'UFR) ?

**Réponse :**

8. Quelle commande lancer depuis le terminal pour attendre les connexions en série (*i.e.* l'une à la suite de l'autre) en IPv6 sur le port 2222 sur la machine `lampe` sous un système Unix (au sein des machines de l'UFR) ?

**Réponse :**

9. Donnez l'écriture big-endian de l'entier dont l'écriture mathématiques en binaire est  $(00011000\ 01000001\ 00000000\ 10100010)_2$ . Pour cela remplissez le tableau de bits ci-dessous où les éléments de la ligne « adresses » représentent des adresses consécutives de 1 bit mémoire de la plus petite adresse (0) à la plus grande (31).

**Réponse :**

adresses	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
valeurs																
adresses	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
valeurs																

10. Que retourne l'instruction `htons(ent)` si `ent` est de type `unsigned short` et la machine, sur laquelle est exécutée l'instruction, est en big-endian ?

**Réponse :**

11. Qu'est-ce que garantit le protocole TCP concernant la communication ?

**Réponse :**

12. A quelle couche du modèle TCP/IP appartient une socket ? Quelles informations une socket regroupe-t-elle ?

**Réponse :**

13. Soit le code suivant :

```
1 int sock = socket(PF_INET6, SOCK_STREAM, 0);
2 if(sock == -1)
3     exit(1);
4
5 struct sockaddr_in6 adrsock;
6 adrsock.sin6_family = AF_INET6;
7 adrsock.sin6_port = htons(1212);
8 if(inet_pton(AF_INET6, ADDR, &adrsock.sin6_addr) <= 0)
9     exit(1);
10
11 int r = connect(sock, (struct sockaddr *) &adrsock, sizeof(adrsock));
```

- (a) Quel type de connexion doit accepter le serveur afin qu'une application exécutant ce code puisse s'y connecter ?

**Réponse :**

- (b) Quel problème de fiabilité pose ce code ? Pourquoi et comment le corriger ?

**Réponse :**

14. On suppose que deux applications communiquent en mode TCP et que l'application 1 exécute le 1<sup>er</sup> bout de code en parallèle de l'application 2 qui exécute le 2<sup>e</sup> bout de code ci-dessous. En supposant que les appels systèmes ne fassent pas d'erreur, l'application 2 affiche-t-elle quelque chose? Quand? Quoi? Comment corriger ce problème?

```
1 //application 1
2 char bufsend[SIZE_MESS];
3
4 sprintf(bufsend, "Bonjour_le_cours_de_programmation_reseaux\n");
5 int ecrit = send(sock1, bufsend, strlen(bufsend), 0);
6 sprintf(bufsend, "Au_revoir\n");
7 ecrit = send(sock1, bufsend, strlen(bufsend), 0);
```

```
1 //application 2
2 char bufrecv[SIZE_MESS+1];
3 memset(bufrecv, 0, SIZE_MESS+1);
4
5 int reçu = recv(sock2, bufrecv, SIZE_MESS, 0);
6 printf("%s\n", bufrecv);
7 reçu = recv(sock2, bufrecv, SIZE_MESS, 0);
8 printf("%s\n", bufrecv);
```

**Réponse :**

15. Soit le code suivant :

```
1 struct addrinfo hints , *r , *p;
2
3 memset(&hints , 0 , sizeof(hints));
4 hints.ai_family = AF_INET6;
5 hints.ai_socktype = SOCK_STREAM;
6
7 if ((getaddrinfo(hostname , port , &hints , &r)) != 0 || r == NULL)
8     exit(1);
9
10 *addrlen = sizeof(struct sockaddr_in6);
11 p = r;
12 while( p != NULL ){
13     *sock = socket(p->ai_family , p->ai_socktype , p->ai_protocol)
14     if(*sock > 0){
15         if(connect(*sock , p->ai_addr , *addrlen) == 0)
16             break;
17
18         close(*sock);
19     }
20     p = p->ai_next;
21 }
22 printf("%d\n" , ntohs(p->ai_addr->sin6_port));
```

- (a) En supposant que les appels systèmes s'exécutent sans erreur, expliquez ce que fait ce code en précisant le type et les valeurs possibles des variables **hostname** et **port**.

**Réponse :**

- (b) Expliquez pourquoi la ligne 16 est nécessaire.

**Réponse :**

- (c) Que faut-il changer dans ce code pour qu'il permette des connexions IPv4 ou IPv6 à un serveur ?

**Réponse :**

16. Soit le code suivant :

```
1 while(1){  
2     int sock2 = accept(sock, NULL, NULL);  
3     if(sock2 == -1) {  
4         exit(1);  
5     }  
6 }
```

- (a) Que représentent les variables `sock` et `sock2` ?

**Réponse :**

- (b) Décrivez les étapes nécessaires à faire avant l'exécution de ce code en précisant les noms des fonctions appelées. Il n'est pas demandé d'écrire du code.

**Réponse :**

- (c) Complétez ce code afin qu'il permette d'accepter plusieurs clients en parallèle en utilisant des processus (non légers) puis d'envoyer à chacun le message « OK ». Les processus zombies doivent être récupérés.

**Réponse :**

17. Donnez les différences entre processus et processus légers.

**Réponse :**

18. Soit le code suivant tel que `serve` soit une fonction de prototype `void *serve(void *)` envoyant un message sur la socket passée en argument :

```
1 int tour = 0;
2
3 while(tour < 10){
4     int sock2 = accept(sock, NULL, NULL);
5     pthread_t thread;
6
7     if (pthread_create(&thread, NULL, serve, &sock2) == -1)
8         continue;
9     tour++;
10 }
```

- (a) Quels problèmes peut-on rencontrer lors de l'exécution de ce code ?

**Réponse :**

- (b) Et comment corriger cela ?

**Réponse :**

19. Soit le code suivant :

```
1 void *serve(void *arg) {
2     int *var = (int *) arg;
3     *var *= 2;
4     int entier = *var;
5     entier++;
6     printf("valeur : %d\n", entier);
7     entier--;
8     int f = open("fic.txt", O_WRONLY | O_APPEND | O_CREAT, 0666);
9     if(f < 0) exit(1);
10    write(f, &entier, sizeof(entier));
11    close(f);
12    return NULL;
13 }
14
```



```

15 int main(int argc, char *argv[]){
16     int tour = 0;
17     int *a = malloc(sizeof(int));
18     *a = 1;
19
20     while(tour < 10){
21         pthread_t thread;
22
23         if (pthread_create(&thread, NULL, serve, a) == -1)
24             continue;
25         tour++;
26     }
27     sleep(10);
28
29     return 0;
30 }

```

- (a) Lors de l'exécution de ce code, que se passe-t-il si l'appel système à `open` en ligne 8 échoue ?

**Réponse :**

- (b) Pourquoi a-t-on mis l'instruction `sleep(10)` en ligne 27 ?

**Réponse :**

- (c) Modifier le programme afin qu'il permette d'obtenir un résultat équivalent sans faire appel à l'instruction `sleep(10)`.

**Réponse :**

(d) Peut-on connaître le contenu du fichier `fic.txt` après exécution du code ? Expliquez

**Réponse :**

(e) Déterminez les sections critiques de ce code.

**Réponse :**

(f) Corrigez le programme, afin qu'après exécution, le fichier `fic.txt` contienne toutes les valeurs prises par la variable `a`, à l'exception de la valeur 1.

**Réponse :**

(g) Donnez alors toutes les valeurs prises par la variable `a`.

**Réponse :**

- (h) Modifiez à nouveau le programme afin qu'il prenne comme unique argument de la ligne de commande le nom du fichier et qu'il passe ensuite la variable `a` et ce nom en paramètre de la fonction `serve` afin qu'elle l'utilise dans l'appel à `open`.

Réponse :

## Prototypes de fonctions pouvant être utiles

```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
int close(int fd);
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
pid_t fork(void);
void freeaddrinfo(struct addrinfo *res);
int listen(int sockfd, int backlog);
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                  void *(*start_routine) (void *), void *arg);
int pthread_join(pthread_t thread, void **retval);
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
const char *inet_ntop(int af, const void *src, char *dst, socklen_t size);
int inet_pton(int af, const char *src, void *dst);
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
int socket(int domain, int type, int protocol);
pid_t waitpid(pid_t pid, int *wstatus, int options);
```