ASR.3.2 TD 12

Applications Client/Serveur
IPv4-versus-IPv6 indépendantes

I. Obtention des adresses de sockets valides pour un host en s'affranchissant des dépendances IPv4-versus-IPv6

`getaddrinfo` et `getnameinfo`

↪ La fonction `getaddrinfo` permet d'obtenir une liste de structures `addrinfo`, chacune contenant une adresse de socket valide pour un nœud et un service spécifiés. Ces adresses de socket pouvent ensuite être données à un appel `bind` ou `connect`, par exemple.

```
struct addrinfo {
               int              ai_flags;
               int              ai_family;
               int              ai_socktype;
               int              ai_protocol;
               socklen_t        ai_addrlen;
               struct sockaddr *ai_addr;
               char            *ai_canonname;
               struct addrinfo *ai_next;
           };
```

On s'intéresse, à titre d'exemple, à l'appel simple :

```
struct addrinfo criteria,res;

getaddrinfo("www.google.fr","http",&criteria,&res);
```

qui demande une liste d'adresses sockets valides pour le nœud "www.google.fr" et le service "http".

Dans l'argument `criteria`, on aura au préalable -si on le souhaite- limiter l'ensemble des adresses sockets retourné, en imposant des critères de sélection. Par exemple :

- `criteria.ai_family = AF_INET6;` pour se limiter aux adresses ipv6 ou alors `criteria.ai_family = AF_UNSPEC;` pour préciser qu'on veut les adresses sockets valides pour le nœud pour toute famille d'adresses.
- `criteria.ai_socktype=SOCK_DGRAM` pour se limiter aux sockets de type non connecté ou alors `criteria.ai_socktype=SOCK_STREAM` pour se limiter aux sockets de type connecté.

↪ La fonction `getnameinfo` permet de convertir une adresse de socket vers les host et service correspondants. Elle permet, en particulier, de remplacer la fonction `inet_ntop` en éliminant les dépendances IPv4-versus-IPv6.

Un exemple d'appel à cette fonction :

```
char host[NI_MAXHOST], service[NI_MAXSERV];

getnameinfo((struct sockaddr *)&sck,sizeof(struct sockaddr_in),host,NI_MAXHOST,
                                                              service,NI_MAXSERV,0);
```

↪ La fonction `gai_strerror` permet de traduire les codes d'erreurs de `getaddrinfo` et de `getnameinfo` en messages destinés à la compréhension humaine.

↪ La fonction `freeaddrinfo` libère la mémoire qui était allouée dynamiquement à la liste chaînée `res`.

II. Restituer le code manquant dans le programme donné ci-dessous. Il prend deux paramètres à la ligne de commande. Le premier est un nom de host ou une adresse ipv4/ipv6 et le deuxième est le nom/numéro d'un service. Le programme affiche alors pour chaque adresse de socket valide, l'adresse ip, le numéro du service, la famille de l'adresse de la socket, le type de socket, le numéro du protocole, la longueur de l'adresse de socket.

Pour les correspondances entre les noms et les numéros, on peut se référer à :

1. `/usr/include/bits/socket.h` pour les familles d'adresses ;
2. `/usr/include/bits/socket_type.h` pour les types de sockets ;
3. `/etc/protocols` pour les protocoles.

```
$ ./a.out www.google.fr 0
216.58.198.3.0
    fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: www.google.fr
216.58.198.3.0
    fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
216.58.198.3.0
    fam: 2 -- socktype: 3 -- pro: 0 -- addrlen: 16 -- canonname: (null)
2a00:1450:4002:801::2003.0
    fam: 10 -- socktype: 1 -- pro: 6 -- addrlen: 28 -- canonname: (null)
2a00:1450:4002:801::2003.0
    fam: 10 -- socktype: 2 -- pro: 17 -- addrlen: 28 -- canonname: (null)
2a00:1450:4002:801::2003.0
    fam: 10 -- socktype: 3 -- pro: 0 -- addrlen: 28 -- canonname: (null)

$ ./a.out "www.google.fr" "http"
216.58.198.195.http
fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: www.google.fr
216.58.198.195.http
fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
216.58.198.195.http
fam: 2 -- socktype: 1 -- pro: 132 -- addrlen: 16 -- canonname: (null)
216.58.198.195.http
fam: 2 -- socktype: 5 -- pro: 132 -- addrlen: 16 -- canonname: (null)
2a00:1450:4007:809::2003.http
fam: 10 -- socktype: 1 -- pro: 6 -- addrlen: 28 -- canonname: (null)
2a00:1450:4007:809::2003.http
fam: 10 -- socktype: 2 -- pro: 17 -- addrlen: 28 -- canonname: (null)
2a00:1450:4007:809::2003.http
fam: 10 -- socktype: 1 -- pro: 132 -- addrlen: 28 -- canonname: (null)
2a00:1450:4007:809::2003.http
fam: 10 -- socktype: 5 -- pro: 132 -- addrlen: 28 -- canonname: (null)

$ ./a.out www.u-pec.fr 0
193.48.143.18.0
    fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: macabeo.u-pec.fr
193.48.143.18.0
    fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
193.48.143.18.0
    fam: 2 -- socktype: 3 -- pro: 0 -- addrlen: 16 -- canonname: (null)

$ ./a.out gatekeeper.iut-fbleau.fr 0
37.58.131.227.0
    fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: gatekeeper.iut-fbleau.fr
37.58.131.227.0
```

```
    fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
37.58.131.227.0
    fam: 2 -- socktype: 3 -- pro: 0 -- addrlen: 16 -- canonname: (null)

$ ./a.out salle225-05.arda 0
172.16.2.78.0
fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: salle225-05.arda
172.16.2.78.0
fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
172.16.2.78.0
fam: 2 -- socktype: 3 -- pro: 0 -- addrlen: 16 -- canonname: (null)

$ ./a.out 172.16.1.101 0
172.16.1.101.0
    fam: 2 -- socktype: 1 -- pro: 6 -- addrlen: 16 -- canonname: 172.16.1.101
172.16.1.101.0
    fam: 2 -- socktype: 2 -- pro: 17 -- addrlen: 16 -- canonname: (null)
172.16.1.101.0
    fam: 2 -- socktype: 3 -- pro: 0 -- addrlen: 16 -- canonname: (null)
--------------------------------------------------------------------------------
1:int main(int argc, char *argv[]){
2:
3:  int r;
4:
5:  struct addrinfo criteria;
6:  struct addrinfo *res,*resp;
7:  char host[NI_MAXHOST], service[NI_MAXSERV];
8:
9:  if (argc < 3){
10:    printf("Usage: %s <host> <service>\n",argv[0]);
11:    exit(1);
12:  }
13:  memset(&criteria,0,sizeof(struct addrinfo));
14:  criteria.ai_family = AF_UNSPEC;  /* Allow IPv4 or IPv6 */
15:  criteria.ai_socktype = 0; //Socket addresses of any type
16:  //The first in the returned list is set to point to the official name of the host
17:  criteria.ai_flags = AI_CANONNAME;
18:  criteria.ai_protocol = 0; /* Any protocol */
19:  r=getaddrinfo(argv[1],argv[2],&criteria,&res);
20:  if (r!=0){
21:      fprintf(stderr, "getaddrinfo fails: %s\n", gai_strerror(r));
22:      exit(EXIT_FAILURE);
23:  }
24:  resp=res;
25:  while (resp!=NULL){
35:  ...........
     .
     .
     .
     ...........
     .
     .
     .
     .
     ..........

     }
     freeaddrinfo(res);
     exit(0);}
```