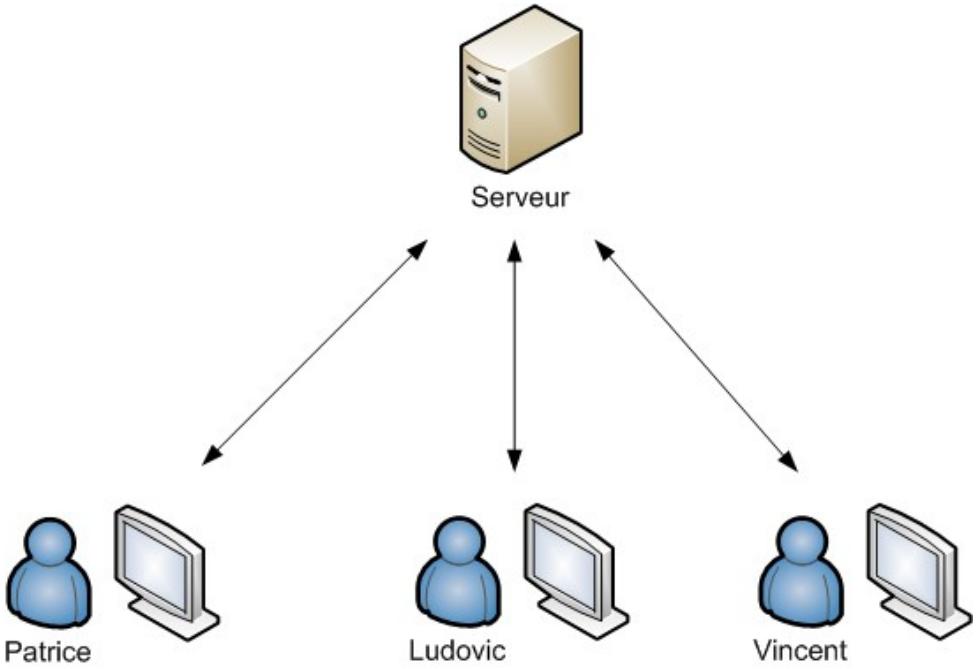
CONNEXIONS TCP/IP en C

D'après un document de Rémy Malgouyres

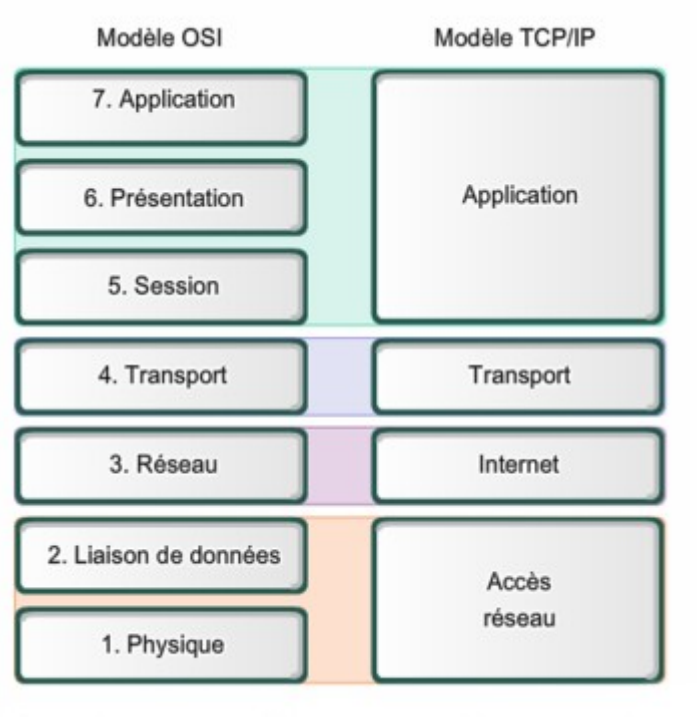
1

Le serveur attend une demande (requete) du client.
Le client effectue une requete.
Le serveur répond.

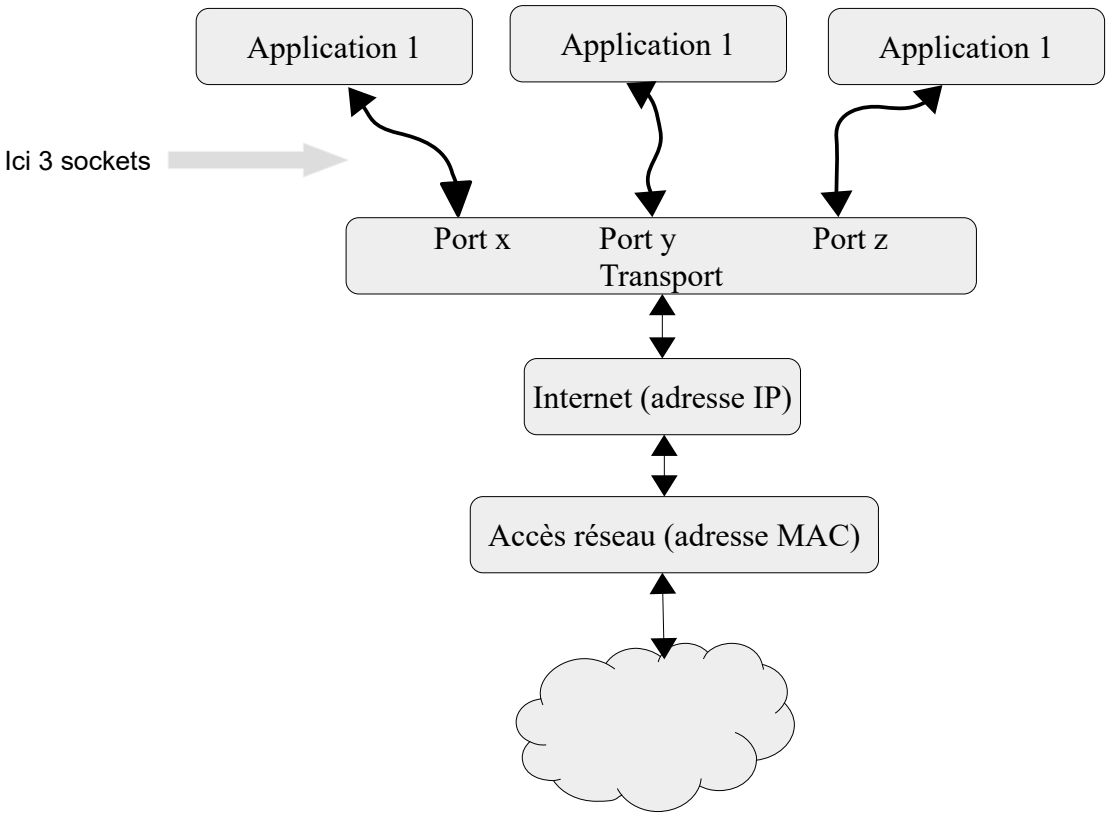
C'est le client qui initiie les échanges



Le système d'exploitation dispose d'un pile logicielle TCP/IP et fournit les supports "Transport" "Internet" "Accès réseau".



Un socket une interface logicielle avec les services du système d'exploitation
Le socket permet d'accéder à un port de la couche transport en ecriture/lecture
Créer un socket revient donc à "brancher" l'application sur un port de la couche transport



2 Création d'un socket en langage C

Les bibliothèques sys/socket.h et netinet/in.h contiennent les fonctions necessaires à l'établissement de la connexion et les déclarations des structures de données d'une connexion.

Structures sockaddr*

Un socket est décrit par:

- un descripteur de fichier (type int) qui servira aux opérations read/write ou recvfrom/sendto selon le protocole TCP ou UDP et par les paramètres de connexion rangés dans une structure

```
#include <netinet/in.h>
struct sockaddr_in {
                  short             sin_family;       // ex AF_INET
                  unsigned short    sin_port;   // ex htons(3490)
                  struct in_addr    sin_addr;
                  char              sin_zero[8];  // en principe mis à 0 avec
                                                  // memset()
                  };

struct in_addr    {
                  unsigned long s_addr;  // charge avec inet_aton()
                  };
```

La fonction socket retourne un identifiant de type int qui servira à désigner le socket.

int sock = socket(AF_INET, SOCK_STREAM, 0);

AF_INET indique une connexion TCP/IP.
SOCK_STREAM pour un mode connecté (TCP)
SOCK_DGRAM pour un socket UDP. (mode non connecté)

Le socket est crée, il faut le lier à une adresse IP et à un port.
Pour cela on initialise une structure sockaddr_in que l'on lie ensuite au socket.

Création d'une structure de type sockaddr_in (on la nomme ici "adresse")
struct sockaddr_in adresse;

Type de connexion
adresse.sin_family = AF_INET;

host-to-network-short. : htons convertit un nombre format réseau en un entier sur 16bits de type hostshort, enregistrable dans la structure de type sockaddr_in

Numero du port (0 pour un port aléatoire)
adresse.sin_port = htons(0);

Choix des adresses écoutées : INADDR_ANY pour toutes
adresse.sin_addr.s_addr = htons(INADDR_ANY);

bind effectue un lien entre un socket et une structure sockaddr. Adresse est "casté" en struct sockaddr.
bind(sock, (struct sockaddr*) &adresse,sizeof(struct sockaddr_in));

CREATION D'UN SOCKET

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>

#define BUFFER_SIZE 256

int cree_socket_tcp_ip()
{
int sock;
struct sockaddr_in adresse;
      if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
      {
            fprintf(stderr, "Erreur socket\n");
            return -1;
      }
      adresse.sin_family = AF_INET;
      adresse.sin_port = htons(0);
      adresse.sin_addr.s_addr = htons(INADDR_ANY);
      if (bind(sock, (struct sockaddr*) &adresse,sizeof(struct sockaddr_in)) < 0)
      {
            close(sock);
            fprintf(stderr, "Erreur bind\n");
            return -1;
      }
      return sock;  // le socket est cree, son numero est retourne
}
```

Remarque :
On peut définir une seule adresse d'écoute avec inet_aton qui remplit la structure sockaddr_in.in_addr
ex : `inet_aton("192.168.0.1", &adresse.sin_addr); // enregiste IP`

3 Affichage de l'adresse d'un socket

getsockname donne l'adresseet le port attribués au socket.
ntohs effectue l'opération inverse de htons et permet de convertir un entier 16bits de type hostshort en un nombre au format compréhensible par l'utilisateur.
inet_ntoa convertit l'adressede type hostshort en une chaine ASCII

```
int affiche_adresse_socket(int sock)
{
struct sockaddr_in adresse;
socklen_t longueur;
      longueur = sizeof(struct sockaddr_in);
      if (getsockname(sock, (struct sockaddr*)&adresse, &longueur) < 0)
      {
            fprintf(stderr, "Erreur getsockname\)n");
            return -1;
      }
printf("IP=%s Port=%u\n",inet_ntoa(adresse.sin_addr),ntohs(adresse.sin_port));
      return 0;
}
```

Réalisation d'un serveur TCP/IP

Il reste à ajouter aux fonctions précédentes la gestion d'un serveur TCP/IP.

Le serveur attend que d'autres programmes sur les machines distantes, appelés clients, entrent en contact avec lui.
On indique que l'on attend une connexion sur le socket crée avec la fonction **listen**.
Le serveur va ensuite attendre un client avec l'appel de la fonction **accept**.
La fonction accept crée une nouvelle socket pour le dialogue avec le client.
Le socket initiale devant rester ouvert et en attente pour la connection d'autres clients.
Le serveur appelle **fork** et crée un processus fils qui va traîter le client, le processus père va boucler à nouveau sur accept dans l'attente du client suivant.

```
int main(void)
{
int sock_contact;
int sock_connectee;
struct sockaddr_in adresse;
socklen_t longueur;
pid_t pid_fils;
      sock_contact = cree_socket_tcp_ip();
      if (sock_contact < 0) return -1;
      listen(sock_contact, 5);
      printf("Mon adresse (sock contact) -> ");
      affiche_adresse_socket(sock_contact);
      while (1)
      {
            longueur = sizeof(struct sockaddr_in);
      sock_connectee = accept(sock_contact,(struct sockaddr*)&adresse,&longueur);
            if (sock_connectee < 0)
            {
                  fprintf(stderr, "Erreur accept\n");
                  return -1;
            }
      pid_fils = fork();
      if (pid_fils == -1)
            {
            fprintf(stderr, "Erreur fork\n");
            return -1;
            }
      if (pid_fils == 0)
            {
            close(sock_contact);
            traite_connection(sock_connectee);  // voir paragraphe suivant
            exit(0);
            }
      else
      close(sock_connectee);
      }
return 0;
}
```

4 Traîtement d'une connexion

Une fois la connexion établie, le serveur connait les données d'adresse IP
et de port du client par la fonction **getpeername**
Le programme dialogue avec le client avec les fonctions read et write.

```
void traite_connection(int sock)
{
struct sockaddr_in adresse;
socklen_t longueur;
```

```
char bufferR[BUFFER_SIZE];
char bufferW[BUFFER_SIZE];
int nb;
      longueur = sizeof(struct sockaddr_in);
      if (getpeername(sock, (struct sockaddr*) &adresse, &longueur) < 0)
            {
            fprintf(stderr, "Erreur getpeername\n");
            return;
            }
sprintf(bufferW, "IP=%s Port=%u\n",
inet_ntoa(adresse.sin_addr),ntohs(adresse.sin_port));
      printf("Connexion : locale (sock_connectee) ");
      affiche_adresse_socket(sock);
      printf(" Machine distante : %s", bufferW);
      write(sock, "Votre adresse : ", 16);
      write(sock, bufferW, strlen(bufferW)+1);
      strcpy(bufferW, "Veuillez entrer une phrase : ");
      write(sock, bufferW, strlen(bufferW)+1);
      nb= read(sock, bufferR, BUFFER_SIZE);
      bufferR[nb-2] = '\0';              // chaine ASCII finit par 0
      printf("L'utilsateur distant a tape : %s\n", bufferR);
      sprintf(bufferW, "Vous avez tape : %s\n", bufferR);
      strcat(bufferW, "Appuyez sur entree pour terminer\n");
      write(sock, bufferW, strlen(bufferW)+1);
      read(sock, bufferR, BUFFER_SIZE);
}
```

5 Le client TELNET

TELNET est un terminel ASCII TCP/IP qui affiche les données reçues du socket sur sa sortie standard et envoie les données saisies depuis son entrée standard vers le socket.
C'est un système client-serveur avec une interface en mode texte pour le client.

L'exemple ci-dessous est une client qui, alternativement :
- lit une chaîne dans le socket et l'affiche sur sa sortie standard ;
- saisit une chaîne sur son entrée standard et l'envoie dans le socket.

Le programme doit être appelé avecdeux paramètres, l'adresse du serveur et le port utilisé.
Ex : christian@ubuntu ~ $ ./client 192.168.1.21 2222

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#define BUFFER_SIZE 256

int cree_socket_tcp_client(int argc, char** argv)
{
      struct sockaddr_in adresse;
      int sock;
      if (argc != 3)
      {
            fprintf(stderr, "Usage : %s adresse port\n", argv[0]);
            exit(0);
```

```
	}
	if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
	{
		fprintf(stderr, "Erreur socket\n");
		return -1;
	}
	memset(&adresse, 0, sizeof(struct sockaddr_in));
	adresse.sin_family = AF_INET;
	adresse.sin_port = htons(atoi(argv[2]));
	if (connect(sock,(struct sockaddr*)&adresse,sizeof(struct sockaddr_in)) < 0)
	{
		close(sock);
		fprintf(stderr, "Erreur connect\n");
		return -1;
	}
	return sock;
}

int affiche_adresse_socket(int sock)
{
struct sockaddr_in adresse;
socklen_t longueur;
	longueur = sizeof(struct sockaddr_in);
	if (getsockname(sock, (struct sockaddr*)&adresse, &longueur) < 0)
	{
		fprintf(stderr, "Erreur getsockname\n");
		return -1;
	}
	printf("IP = %s, Port = %u\n", inet_ntoa(adresse.sin_addr),
	ntohs(adresse.sin_port));
	return 0;
}

int main(int argc, char**argv)
{
int sock;
char buffer[BUFFER_SIZE];
	sock = cree_socket_tcp_client(argc, argv);
	if (sock < 0)
	{
		puts("Erreur connection socket client");
		exit(1);
	}
	affiche_adresse_socket(sock);
	while(1)
	{
		if (read(sock, buffer, BUFFER_SIZE)==0)  break;
		puts(buffer);
		if (fgets(buffer, BUFFER_SIZE, stdin) == NULL) break;
		buffer[strlen(buffer)-1] = '\0';
		write(sock, buffer, BUFFER_SIZE);
	}
	return 0;
}
```

6 Serveur TELNET

Le serveur correspondant est programmé comme le serveur TCP précédent, sauf que la fonction traite_connection doit être modifiée pour tenir compte du fonctionnement du client

(alternance des lectures et écritures dans le socket).

```
void traite_connection(int sock)
{
struct sockaddr_in adresse;
socklen_t longueur;
char bufferR[BUFFER_SIZE];
char bufferW[BUFFER_SIZE];
char tmp[50];
int nb;
      longueur = sizeof(struct sockaddr_in);
      if (getpeername(sock, (struct sockaddr*) &adresse, &longueur) < 0)
      {
            fprintf(stderr, "Erreur getpeername\n");
            return;
      }
      sprintf(bufferW, "IP = %s, Port = %u\n",
      inet_ntoa(adresse.sin_addr),
      ntohs(adresse.sin_port));
      printf("Connexion : locale (sock_connectee) ");
      affiche_adresse_socket(sock, tmp);
      printf(tmp);
      printf(" Machine distante : %s", bufferW);
      strcat(bufferW, "Votre adresse : ");
      affiche_adresse_socket(sock, tmp);
      strcat(bufferW, tmp);

      strcat(bufferW, "Veuillez entrer une phrase : ");
      write(sock, bufferW, BUFFER_SIZE);
      nb= read(sock, bufferR, BUFFER_SIZE);
      printf("L'utilsateur distant a tape : %s\n", bufferR);
      sprintf(bufferW, "Vous avez tape : %s\n", bufferR);
      write(sock, bufferW, BUFFER_SIZE);
}
```