X - Les messages

Les messages

structure

On peut utiliser une structure pour la manipulation interne au programme des données.

```
struct personne {
 uint8_t age;
 uint64_t secu;
 char nom[100];
};

struct personne lire_donnees() {
 struct personne p;
 printf("Entrez votre âge :");
 scanf("%hhu", &(p.age));
 printf("Entrez votre numéro de sécurité sociale :");
 scanf("%lu", &(p.secu));
 printf("Entrez votre nom :");
 scanf("%s", p.nom);

 return p;
}
```

Les messages

structure vers buffer

Il ne faut pas passer directement une structure aux fonctions `send`, `sendto` ou `write` car il peut y avoir des ajouts dus aux problèmes d'alignements.
On écrit donc la structure dans un buffer :

```
int struct2buf(struct personne p, char *buf) {
 int index;
 memcpy(buf, p.nom, strlen(p.nom));
 index = strlen(p.nom);
 buf[index] = 0;
 index += 1;
 memcpy(buf+index, &(p.age),1);
 index += 1;
 memcpy(buf+index, &(p.secu), 8);
 index += 8;

 return index;
}
```

et on récupère la taille en octets des données écrites dans le buffer.

Les messages

envoi

On peut maintenant envoyer les données :

```
int envoi_donnees(int sock, struct personne p, struct sockaddr_in6 adr) {
 char *buf = malloc(109 * sizeof(char));

 int sent, size = struct2buf(p, buf);

 sent = sendto(sock, buf, size, 0, (struct sockaddr*) &adr, sizeof(adr));
 if(sent < 0) return -1;

 free(buf);
 return sent;
}
```

Attention, la variable `buf` ne représente pas une chaîne de caractères ici. On ne peut donc **pas** utiliser `strlen(buf)` pour connaître la taille de `buf`.

Les messages

réception

Pour la réception, si on veut récupérer les données dans une structure, on ne peut pas le faire directement avec `recv`, `recvfrom` ou `read` :

- on récupère les données dans un buffer,
- puis on écrit ces données dans une structure. On peut définir pour cela une fonction `buf2struct`.

```
int reception_donnees(int sock, struct personne *p) {
 char *buf = malloc(109 * sizeof(char));

 rec = recv(sock, buf, sizeof(buf), 0);
 if(rec < 0) return -1;

 buf2struct(p, buf);

 free(buf);
 return 0;
}
```

Les messages

lire un flux de données

En TCP, on a déjà vu qu'il faut une boucle de lecture.

Pour le TP7, il fallait en boucle :

- accumuler dans un buffer les données reçues,
- extraire les lignes du buffer pour les afficher et les examiner, soit en boucle :
  - transformer le buffer en chaîne de caractères,
  - rechercher avec `strstr` la première occurence de la chaîne `''\r\n''`,
  - si on en trouve une : remplacer la caractère `'\r'` par `0`, afficher la ligne, décaler les caractères qui suivent cette chaîne, au début du buffer et mettre à jour la taille du buffer.

Les messages

lire un flux de données

```
int lecture(int sock, char *buf) {
 int cont = 1, taille = 0;
 while(cont) {
  char *fin;
  int lu = recv(sock, buf+taille, BUFSIZE-1-taille, 0);
  if(lu <= 0) return -1;
  taille += lu;
  buf[taille] = 0;
  while((fin = strstr(buf, "\r\n")) != NULL){
   *fin = 0;
   printf("%s\n", buf);
   if(derniere_ligne(buf)) {
    cont = 0;
    break;
   }
   taille -= fin+2-buf;
   memmove(buf, fin+2, taille+1);
  }
 }
 return 0;
}
```