PR6 – Programmation réseaux
TP n° 7 : Un client dict

Exercice 1 : requête DEFINE

1. Nous allons reprendre l'exercice 5 du TP6 en le détaillant.

On rappelle qu'il n'est pas garanti que lorsqu'on appelle la fonction `recv`, on reçoive tout le message envoyé par le serveur. Ni qu'on recevra le contenu d'un seul `send`. En pratique, `send` écrit dans la socket, et `recv` lira au plus `size` octets dans la socket.

Il faudra donc stocker les messages reçus dans un tampon, et chercher dans ce tampon la chaîne de caractères de fin qui nous intéresse.

Pour cela, écrivez une fonction qui va en boucle :

- lire le message sur la socket et le stocker dans le tampon à la suite de ce qu'il contient déjà,
- et pour chaque ligne (*i.e.* terminant par `''\r\n''`) du tampon :
  - afficher la ligne,
  - tester si elle contient la chaîne de fin (*i.e.* ligne commençant par `''250''` ou `''552''`) et terminer dans ce cas,
  - sinon l'ôter du tampon.

Suggestion : définir un type structuré pour le tampon, et la tête de lecture, ainsi que les fonctions suivantes :

```
typedef struct t {
  char buf[2*SIZE_MSG+1];
  int cur;              // position de la ligne suivante dans le tampon
  int size;             // taille courante du tampon
} buf_t;

int find_newline(buf_t* buf);
int is_last_line(buf_t* buf);
```

2. Faites en sorte que les ligne contenant uniquement un point en plus de la chaîne de fin de ligne, s'affiche juste comme un saut de ligne.

Exercice 2 : client `dict`

La commande `MATCH` permet entre autres d'obtenir une liste de mots correspondants à une expression régulière.

1. Testez cette commande avec `nc`.

2. Écrivez un programme qui prend en argument une commande (`SHOW`, `DEFINE`, `MATCH`) et affiche les réponses du serveur.