b) Architecture de type client/serveur : Dans l'exemple ci-dessous, un serveur attend une requête de clients. Le corps de la requête comprend un tableau de caractères et un entier indiquant le PID du processus client ayant émis la requête. Le client place dans le tableau de caractères une chaîne que le serveur doit inverser avant de la renvoyer comme réponse au processus ayant émis la requête. Le type de la réponse est tout simplement le PID du client.

On lance dans un premier terminal le serveur et ce terminal reste bloqué, pour y lire les indications du serveur. Dans un second terminal, on lance un client.

On aurait pour le client :

```
Client operationnel
--> Bonjour
Requete de 11648 envoyee: ->Bonjour<-
11648: reponse du serveur ->ruojnoB<-
-->
```

Et pour le serveur :

```
Serveur: requete ->Bonjour<- du processus 11648
Serveur: tmp=7
Serveur: rep.chaine ->ruojnoB<-
Serveur: reponse envoyee a 11648
```

Programmer l'application ci-dessus, en faisant preuve de créativité (présentation, forme des messages ,…) et en faisant en sorte que la communication se fasse en utilisant une file de messages et/ou une mémoire partagée.

Puis programmer une autre application avec un client avec un mode de communication de votre choix qui adresse des questions au serveur qui répond aux questions et affiche les réponses. Les questions sont des opérations mathématiques simples à effectuer (par exemple « combien font 7 * 13 ? »). On commencera par définir la structure contenant les questions, c'est-à-dire une opération (+,-,* et /) et deux opérandes.

A rendre:

- un readme expliquant la démarche que vous avez employée pour résoudre ce problème (sous-programme, algorithmes utilisés, …); et comment compiler votre programme.

- un programme en langage C (avec des procédures et/ou des fonctions et qui est commenté) permettant de mettre en œuvre la solution à ce problème. Les moyens de communication utilisé doivent être expliqués.