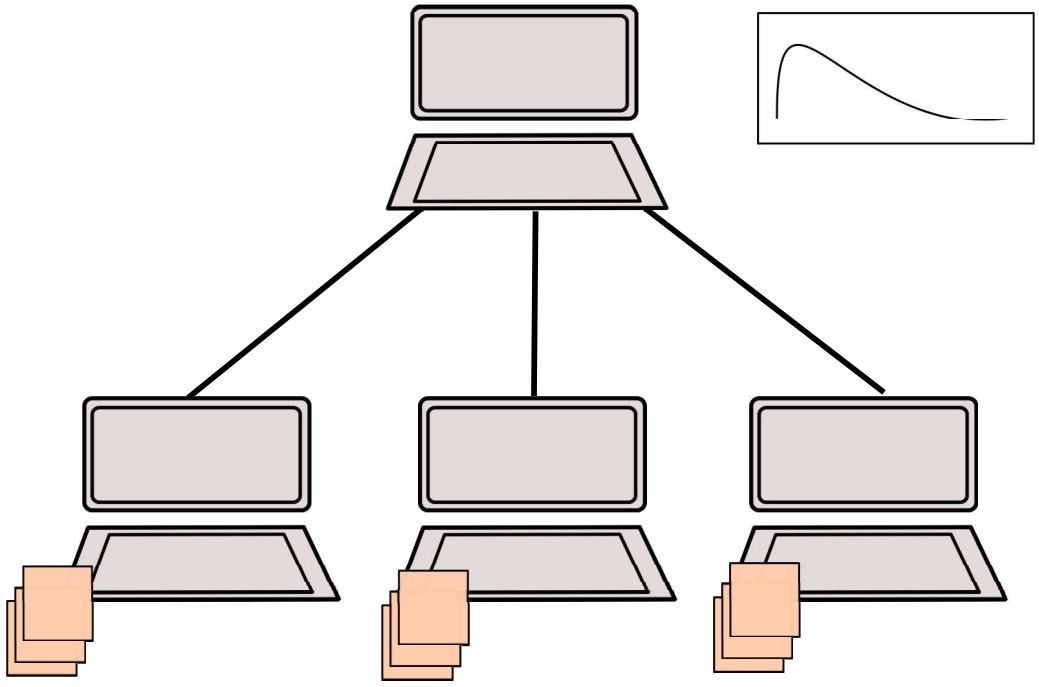
# Projet: Algorithmes et C appliqués aux Systèmes Numériques

**John Samuel**

## Année: 2019-2020

## Projet

L'objectif du projet est de créer un environnement client-serveur pour l'analyse et la gestion des images et de leurs métadonnées. Cependant, contrairement aux applications client-serveur traditionnelles, nous n'enverrons pas les images au serveur pour analyse, mais nous effectuerons une analyse d'image côté client et enverrons des rapports de synthèse au serveur.



## Client-Serveur

Un environnement client-serveur permet un mode de communication sur un réseau entre les programmes. Un de ces programmes est appelé serveur qui répond aux requêtes d'autres programmes appelés clients. Par exemple,

1. Un client envoie un message au serveur et le serveur reçoit le message et

renvoie.
2. Un client envoie l'opérateur et un/deux numéros et le serveur fait le calcul et envoie le résultat.

Référence : https://fr.wikipedia.org/wiki/Client-serveur

Dans ce projet, le serveur connaît uniquement les informations suivantes pour chaque client et stocke ces informations.

- Nom de la machine cliente
- Nombre d'images sur l'ordinateur client
- Balises d'image distinctes sur l'ordinateur client (par exemple, chats, chiens, etc.)
- Couleurs prédominantes des images sur la machine cliente (par exemple, # 2020DE, # FF3E23, etc.)

Il y a trois tâches dans ce projet. Dans notre première tâche, votre mission consiste à envoyer différents types de messages.

1. Le client envoie son nom et le serveur renvoie le même nom, en guise d'accusé de réception.
2. Un simple message envoyé par le client, auquel le serveur renvoie une réponse.
3. Le client envoie deux numéros et un opérateur mathématique et le serveur répond le résultat de l'opération.
4. Le client envoie N couleurs et le serveur les enregistre dans un fichier.

## Tâche 1.

#### a. message

Pour simuler un environnement client-serveur, on va utiliser une seule machine en lançant deux terminaux. Sur un des deux terminaux exécutez le code du serveur. Sur l'autre, exécutez le code du client.

Téléchargez les fichiers suivants à partir d'e-campus : client.h, client.c, serveur.h, serveur.c, Makefile. Lisez bien tous les fichiers. Exécutez

```
$ make
```

et voyez les fichiers exécutables qui sont créés.

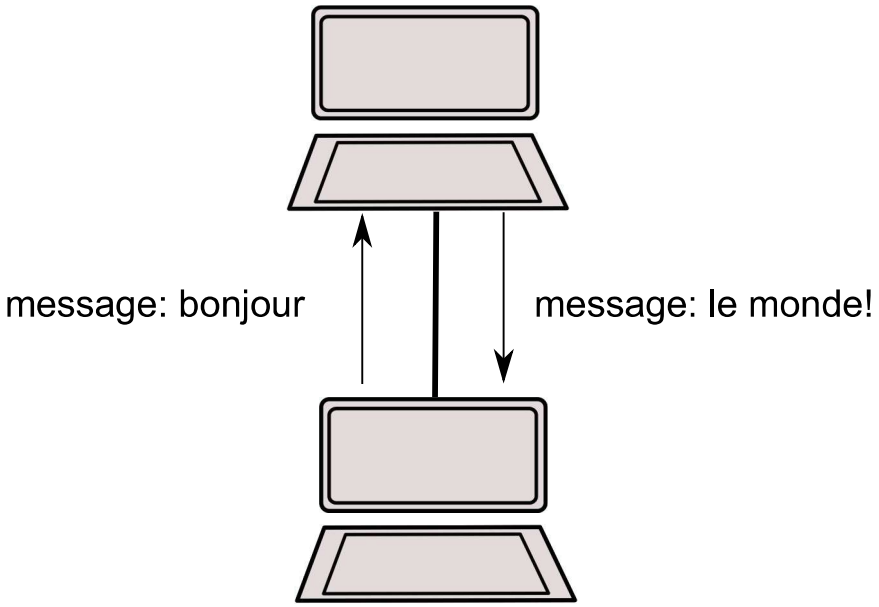
Ouvrez deux terminaux. Sur le premier terminal, exécutez

```
./serveur
```

et sur le second terminal

```
./client
```

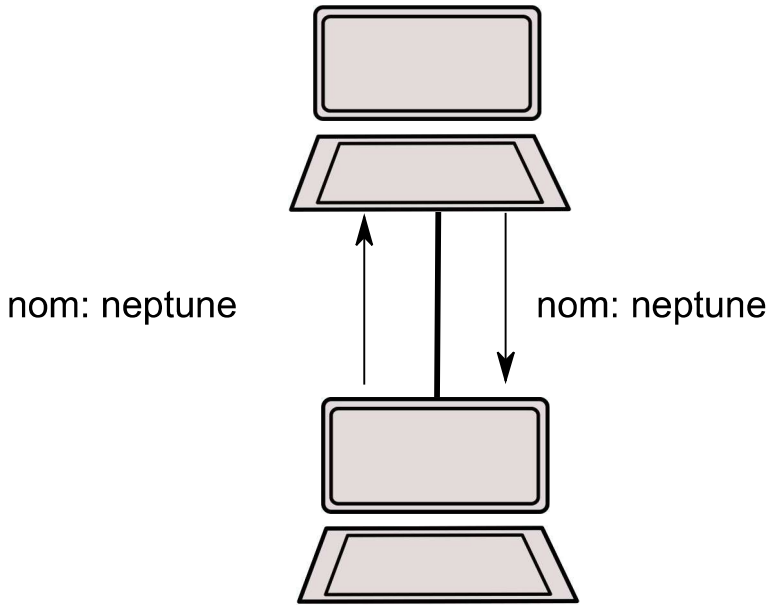
Entrez un message et voyez les affichages sur les deux terminaux.




Modifiez la fonction recois_envoie_message (serveur.c). Quand le serveur reçoit un message, il demande à l'utilisateur de saisir un message et envoie ce message au client. Testez votre code. N'oubliez pas d'utiliser make (pour la compilation et la génération des fichiers exécutables).

**b. nom**

Vous avez remarqué les premiers caractères dans chaque message commencent par 'message' et suivi par :. Pour toutes les prochaines missions, nous devons remplacer "message" par "nom"/"calcul"/"couleurs". Votre prochain objectif est d'écrire une fonction envoie_nom_de_client(...) dans le fichier client.c (C'est à vous pour décider le nom) et une fonction renvoie_nom(...) dans le fichier serveur.c qui renvoie le nom.



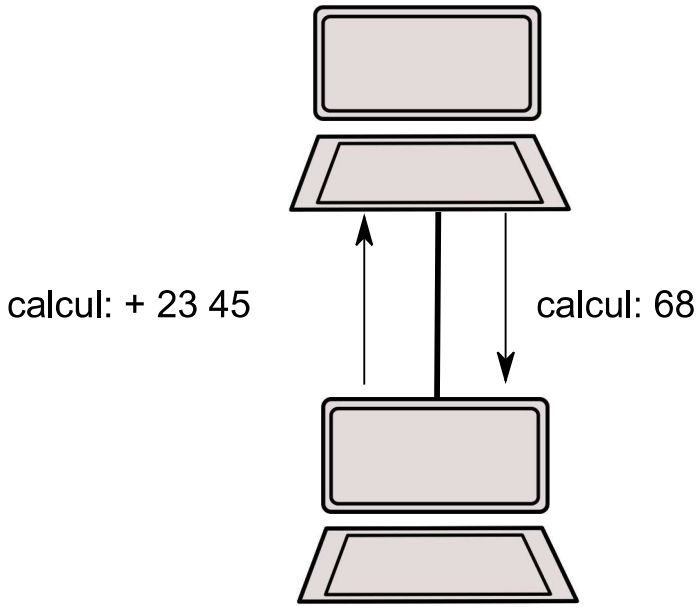

**c. calcul**

Votre prochaine mission consiste à effectuer des opérations mathématiques côté serveur. Modifiez les fichiers client.c et serveur.c pour prendre en charge des opérations mathématiques simples (+,-,*,...). Ajoutez une fonction envoie_operateur_numeros(...) dans le fichier client.c et recois_numeros_calcule(...) dans le fichier serveur.c. Le client envoie l'opérateur et un (ou deux) numéros et le serveur envoie le résultat. Par exemple, si le client envoie le message.

```
calcule : + 23 45
```

Le serveur répond

```
calcule : 68
```



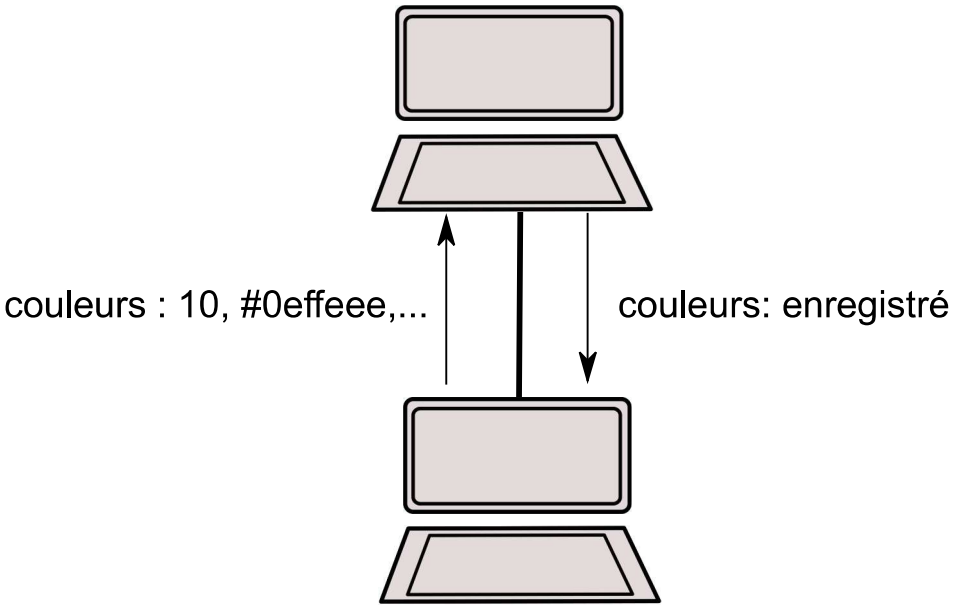

Testez votre code avec des nombres à virgule flottante et des entiers.

#### d. couleurs

Votre dernière mission dans cette tâche consiste à écrire une fonction envoie_couleurs(...) dans le fichier client.c et une fonction recois_couleurs(...) dans le fichier serveur.c. Le client envoie N couleurs (N<30) en utilisant le codage RGB (https://fr.wikipedia.org/wiki/Rouge_vert_bleu) et le serveur reçois ces couleurs et les enregistre dans un fichier.

par exemple, si le client veut envoyer 10 couleurs, il enverra le message suivant

```
couleurs: 10, #0effeee,...
```


couleurs : 10, #0effeee,...
couleurs: enregistré