MINISTERE DE L'ENSEIGNEMENT SUPERIEURET
DE LA RECHERCHE SCIENTIFIQUE UNIVERSITE
TUNIS EL MANAR
INSTITUT SUPERIEUR D'INFORMATIQUE

# Rapport de mini projet programmation sous Unix

## Communication client/serveur avec les tubes nommés et les sockets

# Liste des figures

# Sommaire

## Contents

# Introduction Générale

Nous souhaitons réaliser un système de communication **client/serveur**. L'architecture de ce système correspond au schéma général suivant : Un serveur attend des questions de clients, une question correspond à la demande d'envoi de n nombres tirés au sort par le serveur (n est un nombre aléatoire compris entre 1 et **NMAX** tiré au sort par le client).

Dans sa question, le client envoi également son numéro de telle sorte que le serveur peut le réveiller quand il a écrit la réponse.
En effet plusieurs clients pouvant être en attente de réponse. Il est nécessaire de définir un protocole assurant que chaque client lit les réponses **qui** lui sont destinées.

# Chapitre 1

# Environnement du projet

# Introduction

Dans ce chapitre, on va mettre le projet dans son cadre général, définir la méthodologie le plus approprié à le projet.

# La méthodologie de conception

En ingénierie, une méthode d'analyse et de conception est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client..

**Notre choix :**

Pour bien réussir notre projet, nous avons opté pour le modèle en-cascade. Ce modèle offre une structure hiérarchique claire pour les projets de développement dans laquelle les différentes phases du projet sont clairement délimitées. Étant donné que chaque phase se termine par une étape, le processus de développement peut être suivi facilement.

# Les phases du modèle en cascade

Dans le modèle en cascade, les différentes phases d'un processus de développement s'enchaînent. Chaque phase se termine par un résultat intermédiaire (étape), par exemple avec un catalogue d'exigences sous la forme d'un cahier des charges, avec la spécification d'une architecture logicielle ou avec une application au stade alpha ou bêta.

**Analyse**

Chaque projet logiciel commence par une phase d'analyse comprenant une étude de faisabilité et une définition des besoins. Les coûts, le rendement et la faisabilité du projet logiciel sont estimés lors de l'étude de faisabilité. Celle-ci permet de créer un cahier des charges (une description grossière des besoins), un plan de projet, une budgétisation du projet et, le cas échéant, un devis pour le client.

Les besoins sont ensuite définis de façon détaillée. Cette définition comprend une analyse réelle et un concept cible. Alors que les analyses réelles décrivent les problèmes, le concept cible permet de définir quelles fonctionnalités et quelles propriétés le produit logiciel doit offrir afin de répondre aux besoins. La définition des besoins permet notamment d'obtenir un cahier des charges, une description détaillée de la façon dont les exigences du projet doivent être remplies ainsi qu'un plan pour le test d'acceptation.

Enfin, la première phase du modèle en cascade prévoit une analyse de la définition des besoins, au cours de laquelle les problèmes complexes sont décomposés en sous-tâches de moindre ampleur et des stratégies de résolution correspondantes sont élaborées.

**Conception**

La phase de conception sert à l'élaboration d'un concept de résolution concret sur la base des besoins, des tâches et des stratégies déterminées au préalable. Au cours de cette phase, les développeurs élaborent l'architecture logicielle ainsi qu'un plan de construction détaillé du logiciel et se concentrent ainsi sur les éléments concrets tels que les interfaces, les frameworks ou les bibliothèques. Le résultat de la phase de conception inclut un document de conception avec un plan de construction logicielle, ainsi que des plans de test pour les différents éléments.

**Implémentation**

L'architecture logicielle élaborée pendant la phase de conception est réalisée lors de la phase d'implémentation qui comprend la programmation du logiciel, la recherche d'erreurs et les tests de modules. Lors de la phase d'implémentation, le projet de logiciel est transposé dans la langue de programmation souhaitée. Les différents composants logiciels sont développés séparément, contrôlés dans le cadre de tests de modules et intégrés étape par étape dans le produit global. Le résultat de la phase d'implémentation est un produit logiciel qui sera testé pour la première fois en tant que produit global lors de la phase suivante (test alpha).

**Test**

La phase de test comprend l'intégration du logiciel dans l'environnement cible souhaité. En règle générale, les produits logiciels sont tout d'abord livrés à une sélection d'utilisateurs finaux

sous la forme d'une version bêta (bêta-tests). Il est alors déterminé si le logiciel répond aux besoins préalablement définis à l'aide des tests d'acceptation développés lors de la phase d'analyse. Un produit logiciel ayant passé avec succès les bêta-tests est prêt pour la mise à disposition.

Après avoir réussi la phase de tests, le logiciel est mis en production pour exploitation. La dernière phase du modèle en cascade inclut la livraison, la maintenance et l'amélioration du logiciel

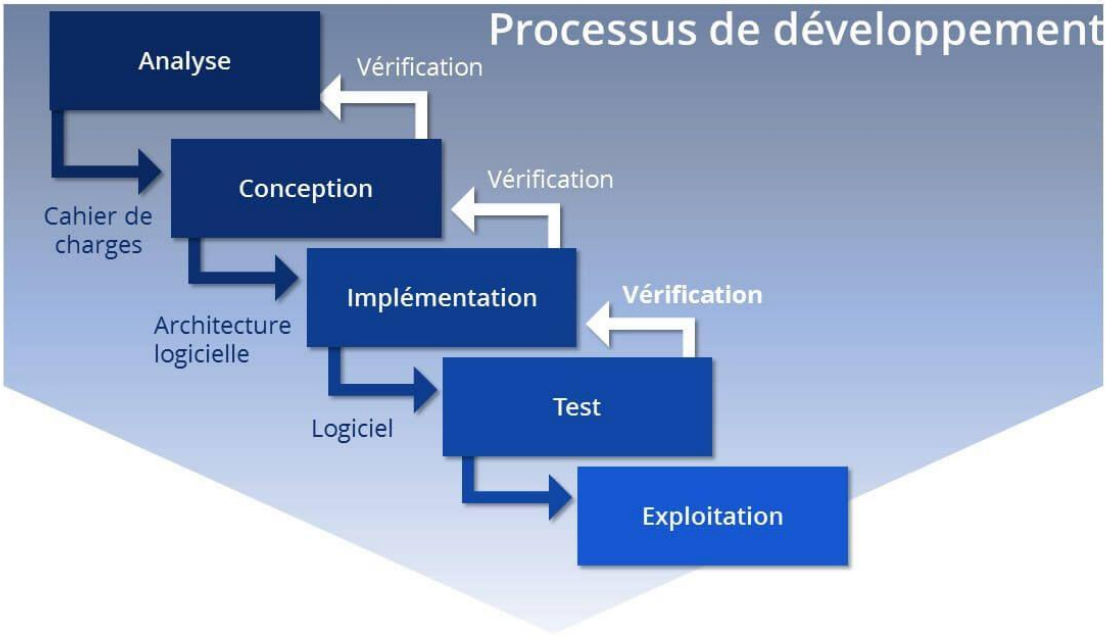



*Figure 1 cascade*

## Conclusion

Dans ce chapitre, on a présenté le projet et la méthodologie de conception. Dans le chapitre suivant, on détailler les différents besoins de ce projet.

# Chapitre 2
# Spécifications des besoins

## Les besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités du système. Ce sont les nécessités spécifiant un comportement d'entrée / sortie du système. En effet, le système permet :

# Partie 1 : Tubes Nommés (Fifo)

### 1- Client

- Ouvrir les tubes nommés
- Construire la question
- Envoyer la question au serveur
- Attendre la réponse du serveur
- Réveiller suite à la réception du signal du serveur « SIGUSR1 »
- Lire la réponse
- Envoyer un signal de confirmation au serveur
- Traiter la réponse localement

### 2- Serveur

- Créer les tubes nommés
- Ouvrir les tubes nommés
- Attendre une question
- Lire une question
- Construire la réponse
- Envoyer la réponse au client
- Envoyer un signal « SIGUSR1 » au client

# Partie 2 : sockets en mode connecté (TCP)

## 1- Client

- Création du socket
- Préparation de l'adresse locale
- Attachement du socket localement
- Demande de connexion au serveur
- Construire la question
- Envoyer la question au serveur
- Attendre la réponse du serveur
- Lire la réponse
- Fermeture de la Socket

## 2- Serveur

- Création du socket d'écoute
- Préparation de l'adresse locale
- Attachement du socket localement
- Attendre la réception des demandes de connexions
- Accepter la demande de connexion d'un client
- Création d'un processus pour la gestion d'un client
- Attendre une question
- Lire une question
- Construire la réponse
- Envoyer la réponse au client
- Fermeture des sockets

# Partie 3 : sockets en mode non connecté (UDP)

## 1- Client

- Création du socket
- Préparation de l'adresse locale
- Attachement du socket localement
- Construire la question
- Envoyer la question au serveur
- Attendre la réponse du serveur
- Lire la réponse
- Fermeture de la Socket

## 2- Serveur

- Création du socket pour le serveur
- Préparation de l'adresse locale
- Attachement du socket localement
- Attendre une question
- Lire une question
- Construire la réponse
- Envoyer la réponse au client
- Fermeture des sockets

## Les besoins non fonctionnels

- **Disponibilité**

  Le serveur doit être toujours disponible et en service pour interagir avec les clients à n'importe quel moment donné.

- **Performance**

  Le temps de réponse du client est crucial, l'interprétation des questions doit donc être rapide et le temps de réponse doit être fluide.

- **Fiabilité**

  Le système devra avoir un fonctionnement fiable et sécurisé assurant l'intégrité des données

- **Facilité d'utilisation (utilisabilité)**

  - ✓ Facilité de compréhension.
  - ✓ Facilité d'apprentissage.

## Conclusion

Au cours de ce chapitre, on a précisé les besoins fonctionnelles et non fonctionnelles du Notre système. Dans le chapitre suivant, on détailler la conception logique pour le projet.

# Chapitre 3

# Conception

# Introduction

Maintenant qu'on a déterminé les besoins exigences du projet, au cours de ce chapitre on détaille la conception logique pour le projet.

# Partie 1 : Tubes Nommés (Fifo)

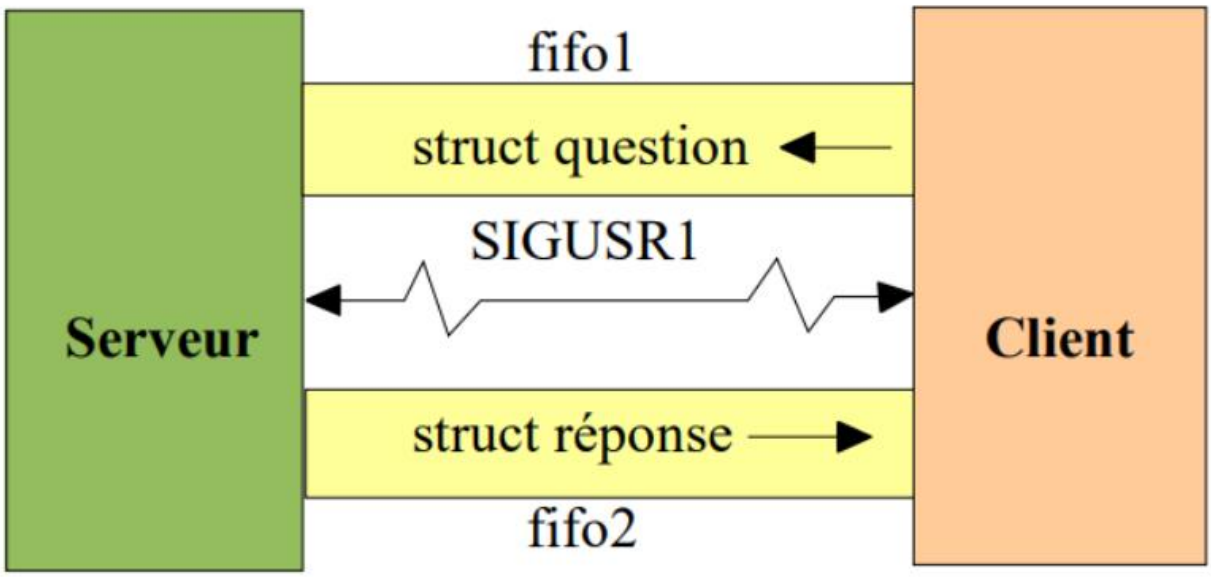



*Figure 1 Fifo*

**Les sockets**

Les sockets sont des flux de données, permettant à des machines locales ou distantes de communiquer entre elles via des protocoles.

Les différents protocoles sont TCP qui est un protocole dit « connecté », et UDP qui est un protocole dit « non connecté ».

# Partie 2 : sockets en mode connecté (TCP)

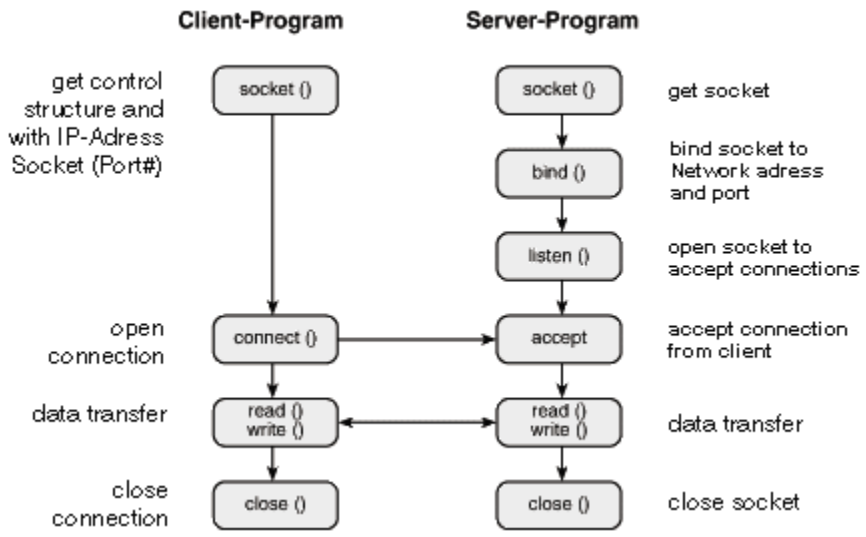



*Figure 2 TCP*

# Partie 3 : sockets en mode non connecté (UDP)

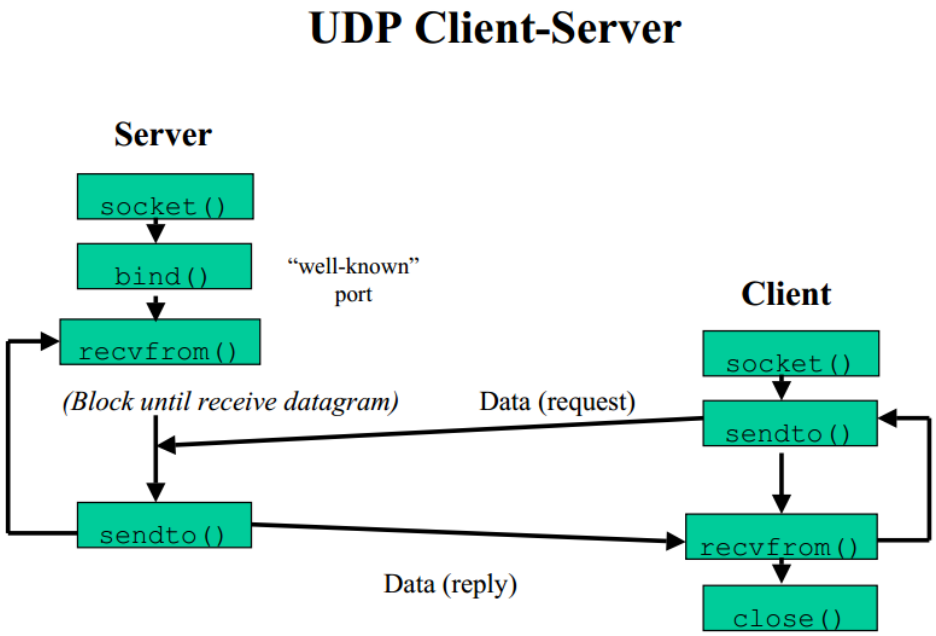



*Figure 3 UDP*

# Conclusion

Au cours de ce chapitre, on a présenté la solution conceptuelle pour le système utilisant la modélisation dynamique. Dans le chapitre suivant, on va décrire le développement du projet.

# Chapitre 4

# Réalisation

# Environnement de développement

## L'environnement matériel

- **Laptop Lenovo ideapad 3**

Processeur : Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz  2.11 GHz

RAM : 12GB

## L'environnement logiciel

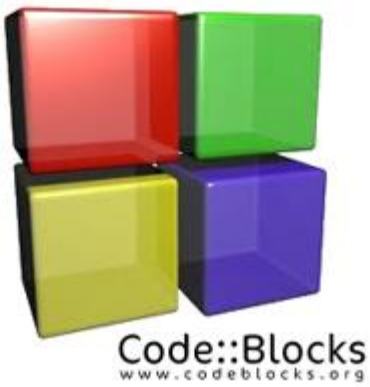
- **Windows 10 Pro**


Windows 10 est un système d'exploitation de la famille Windows NT développé par la société américaine Microsoft. Officiellement présenté le 30 septembre 2014

- **VMware Workstation**

VMware Workstation est un outil de virtualisation de poste de travail créé par la société VMware, il peut être utilisé pour mettre en place un environnement de test pour développer de nouveaux logiciels, ou pour tester l'architecture complexe d'un système d'exploitation avant de l'installer réellement sur une machine physique.

- **Code blocks**




Code Blocks est une excellente option pour la programmation en C. Il s'agit d'un environnement de développement multiplateforme open source intégrée qui supporte l'utilisation de compilateurs multiples

## Interfaces

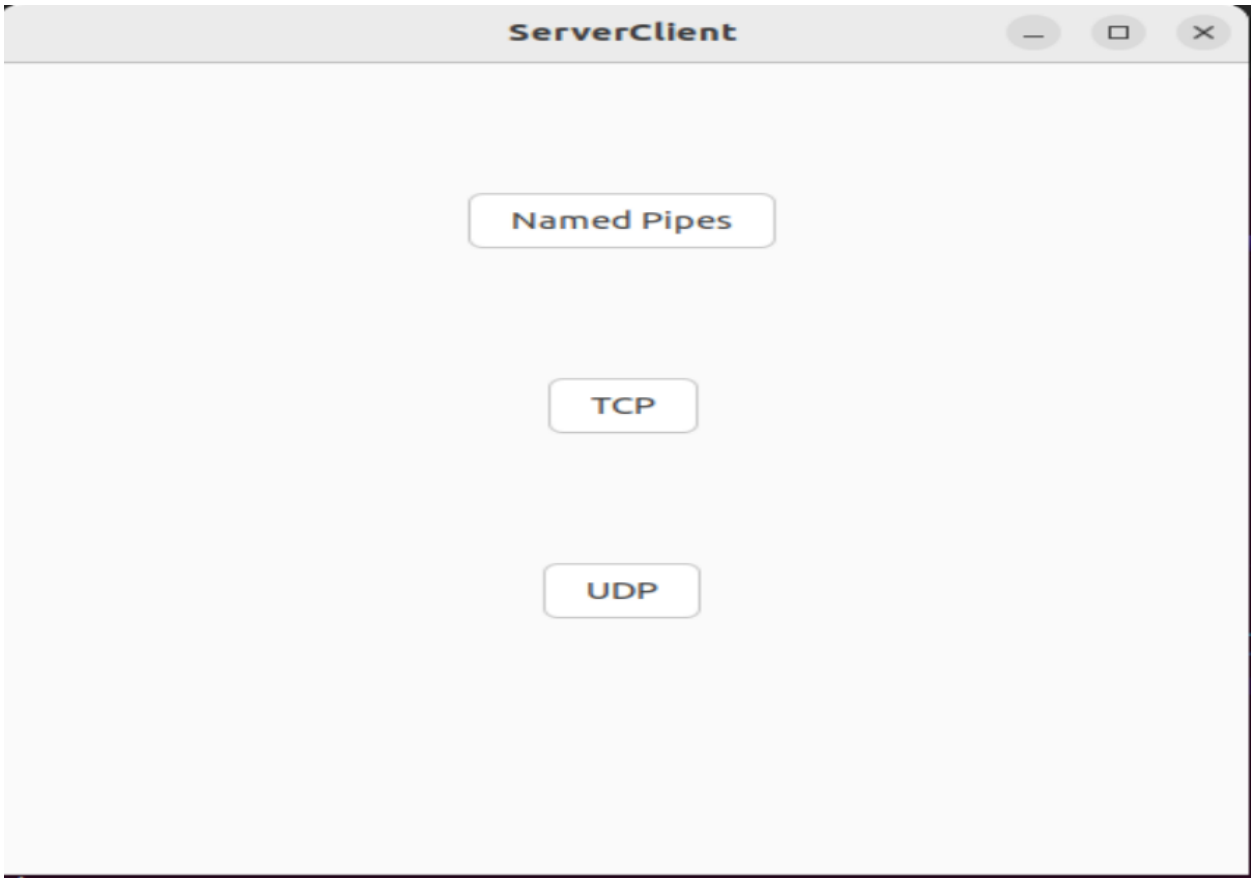
Après la génération de l'exécutable par la commande Shell :

```
gcc -o ServerClient ClientServer.c -Wall pkg-config --cflags --libs gtk+-3.0 -export-dynamic
```

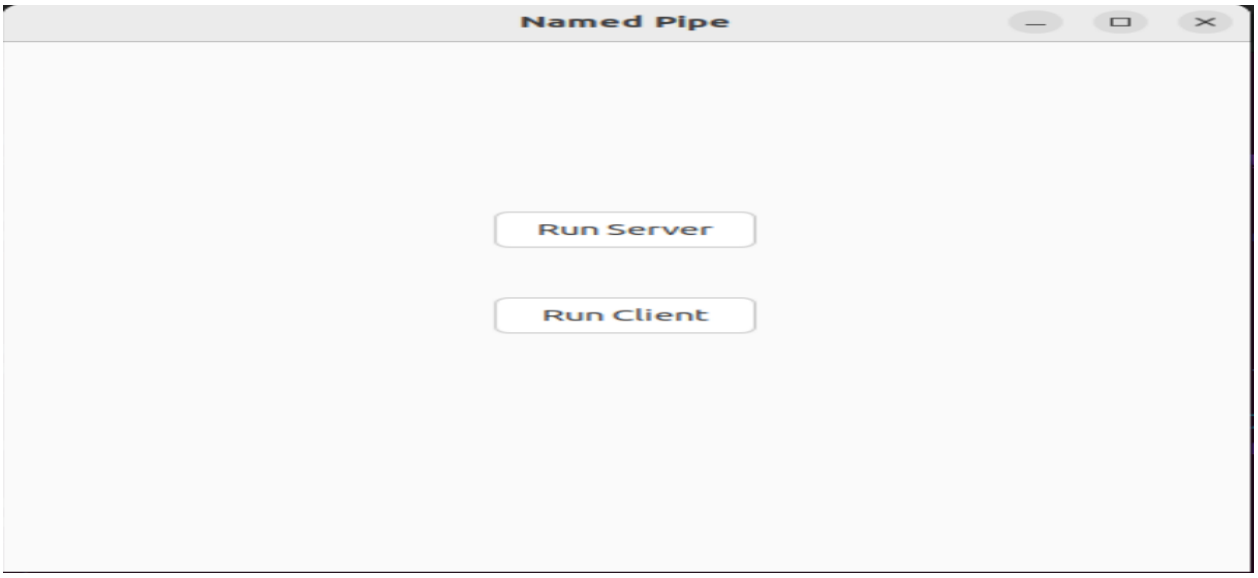
On va exécuter l'exécutable généré par la commande :

```
./ServerClient
```

Par conséquent l'interface de choix ci-dessous se lance

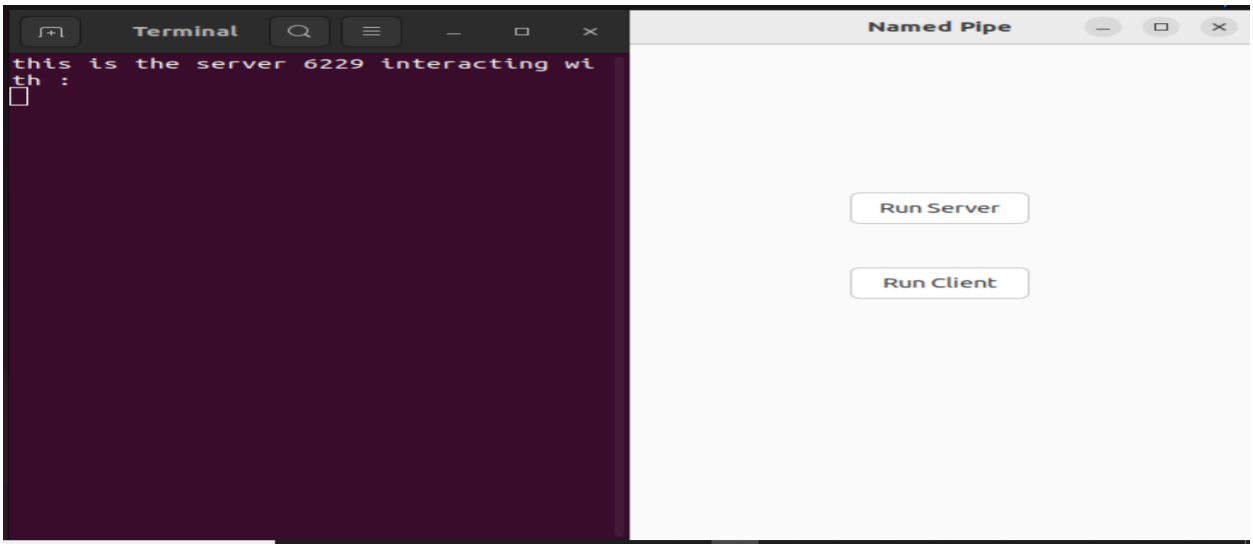



Cette interface donne le choix à l'utilisateur de choisir la méthode d'exécution des processus client-serveur ; si en utilisant les tubes nommés , si en utilisant les sockets en mode connecté TCP , si en utilisant les sockets en mode non connecté UDP .

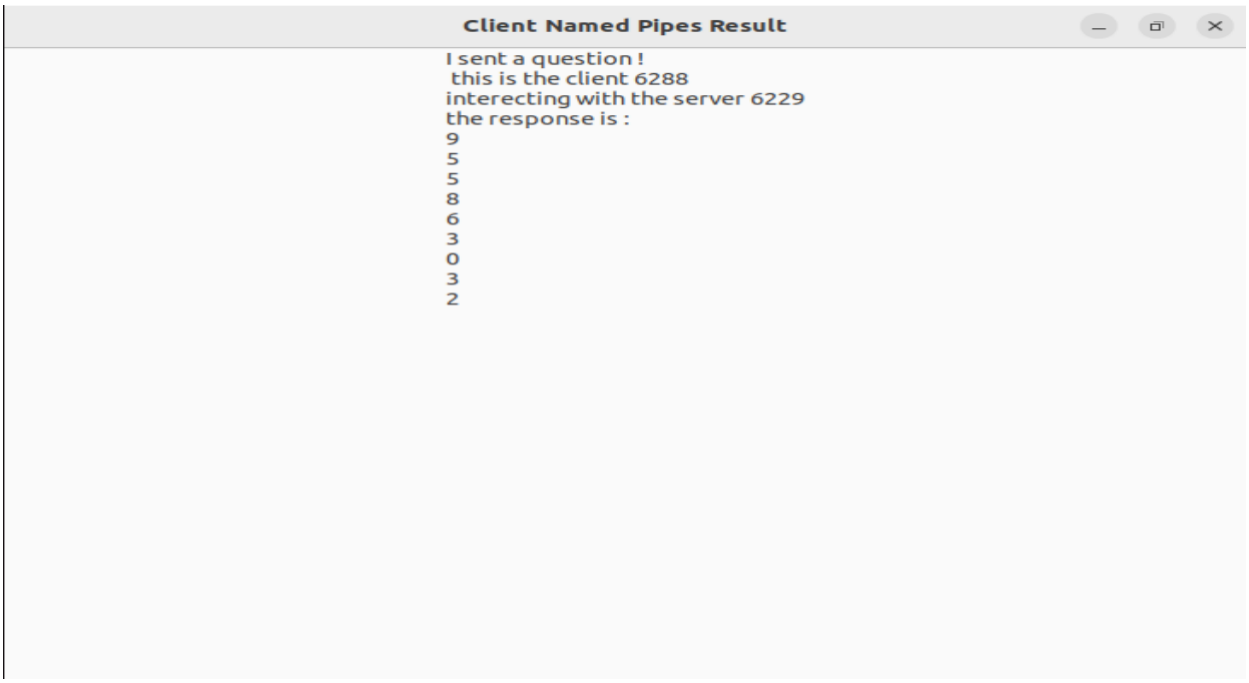



Cette interface se lance lors du choix de l'utilisateur des tubes nommées en cliquant sur le bouton « Named Pipes » .

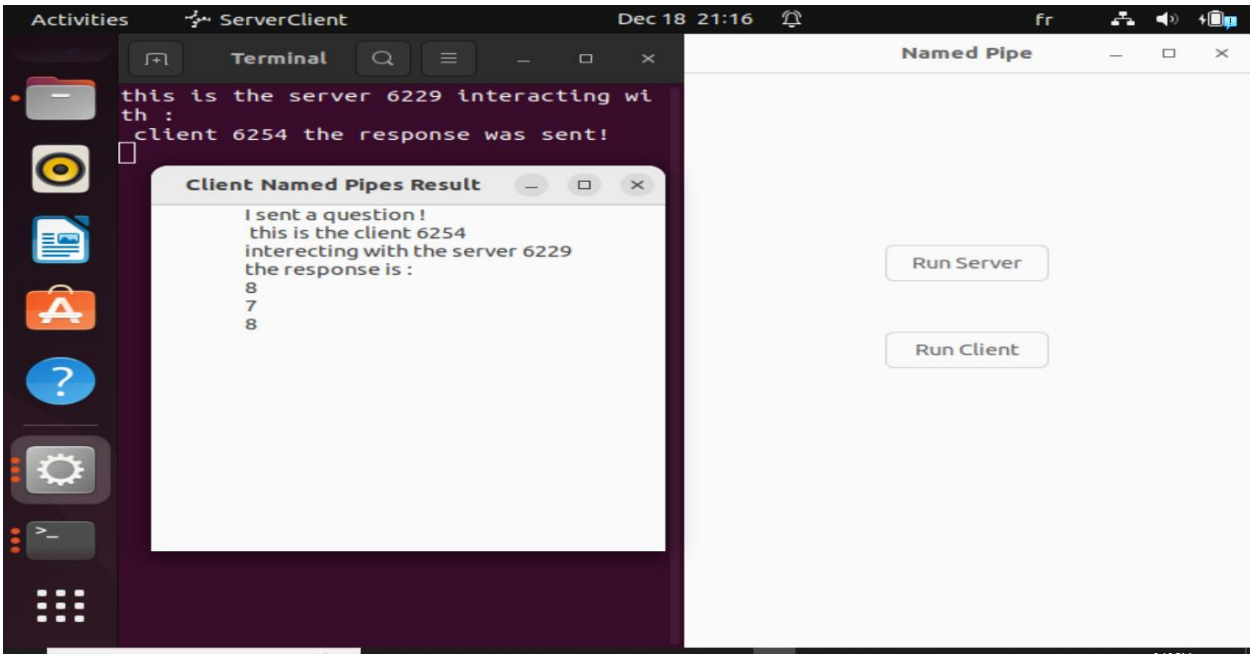
Les deux boutons visualisé le « Run Server » met le serveur en exécution et l'autre « Run Client » met le client en exécution tout en nottant que le serveur est multi-clients donc on peut lancer plusieurs clients .



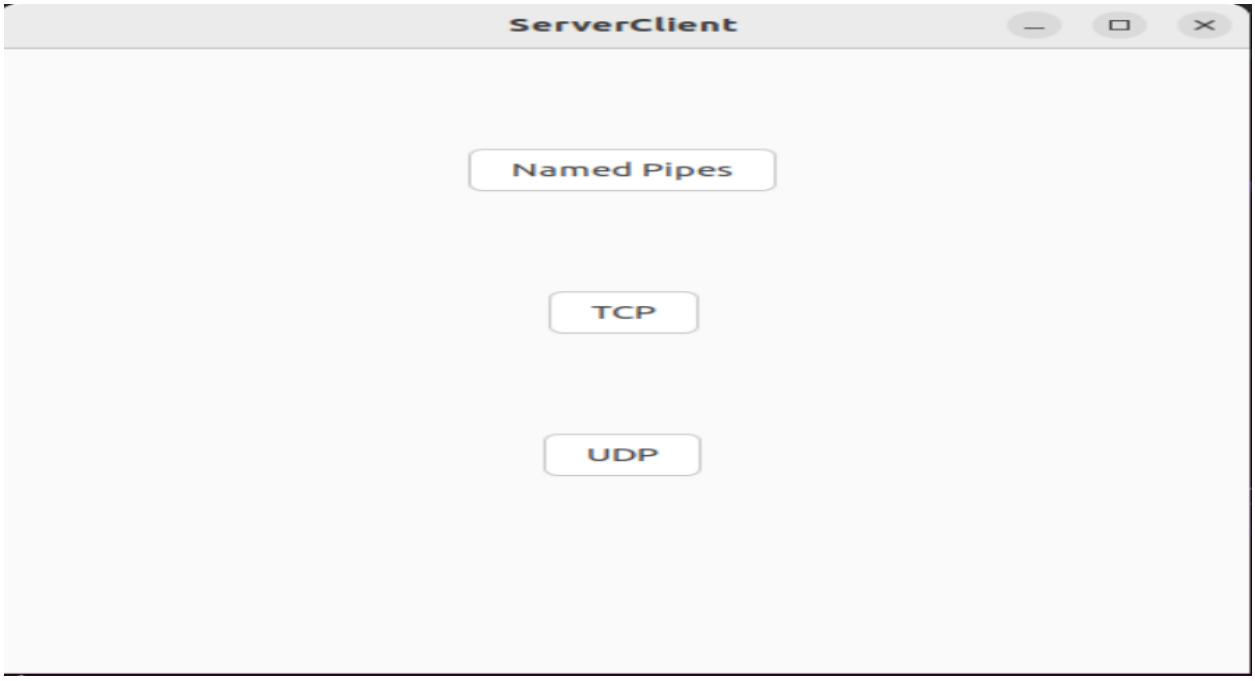

Le bouton "Run Server" permet au serveur de se lancer en terminal et reste ouvert en attendant les clients ou un signal kil



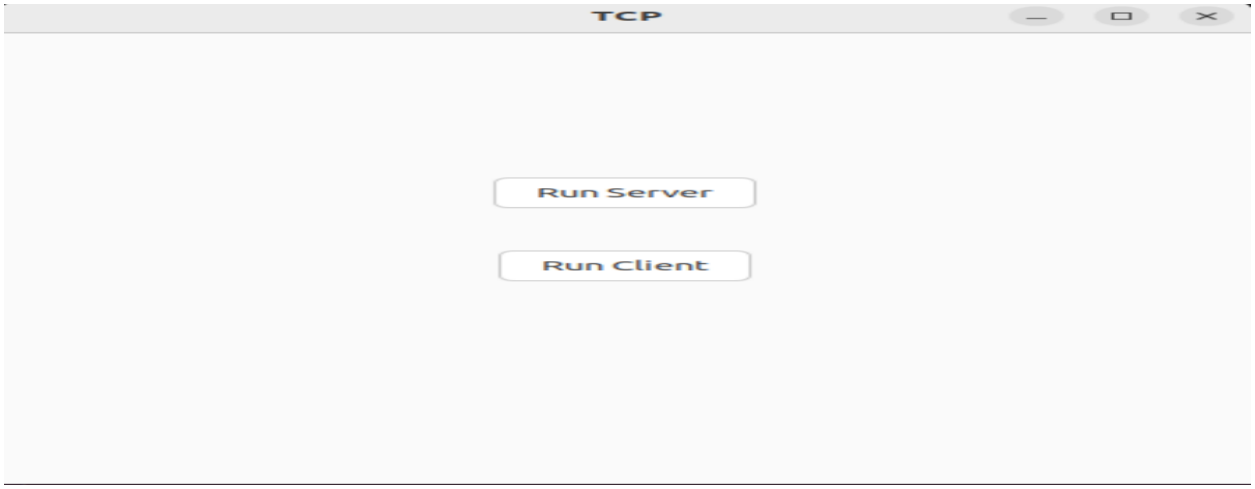

Le bouton “Run Client” génère cette interface qui donne le résultat envoyé en réponse de serveur a la question du client.



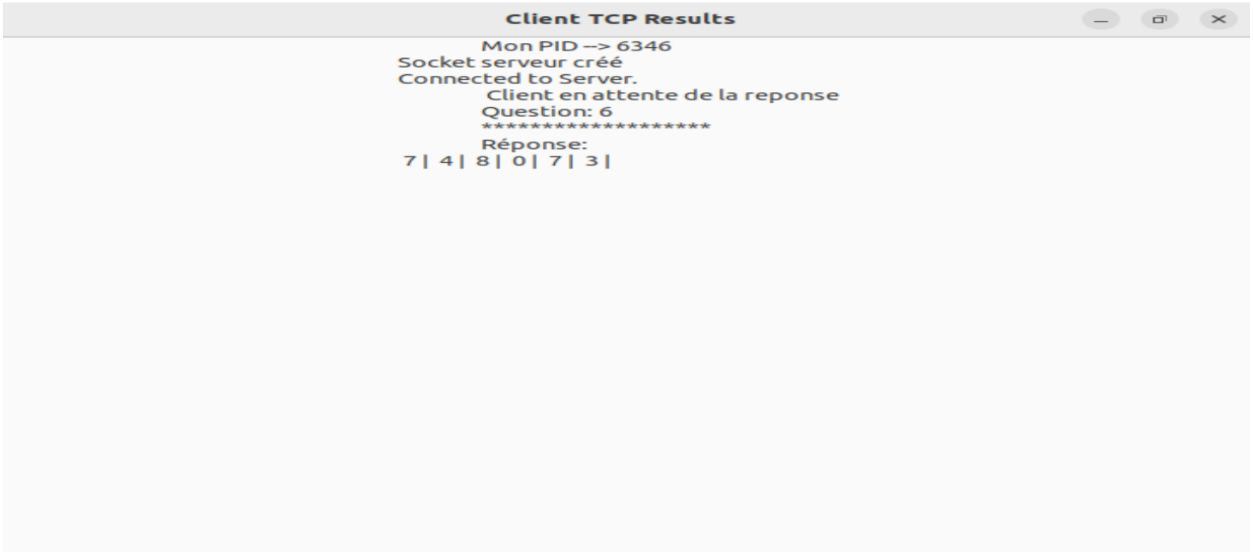

La trace d’exécution.



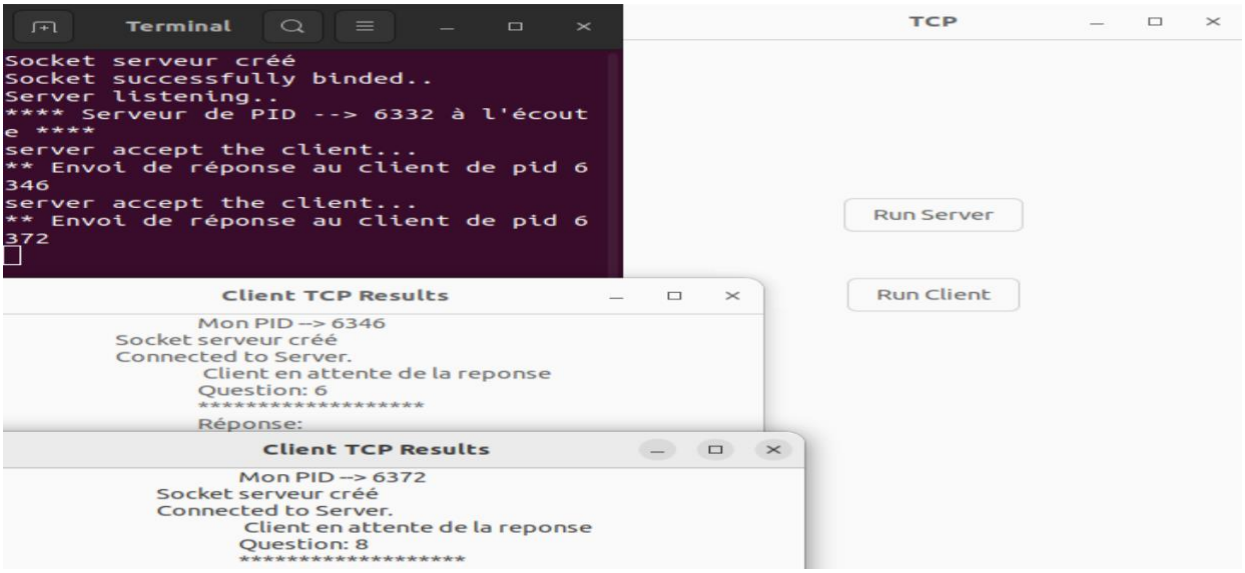

Cette interface se lance lors du choix de l'utilisateur du mode connecté TCP en cliquant sur le bouton « TCP ».
Les deux boutons visualisé le « Run Server » met le serveur en exécution et l'autre « Run Client » met le client en exécution tout en nottant que le serveur est multi-clients donc on peut lancer plusieurs clients.



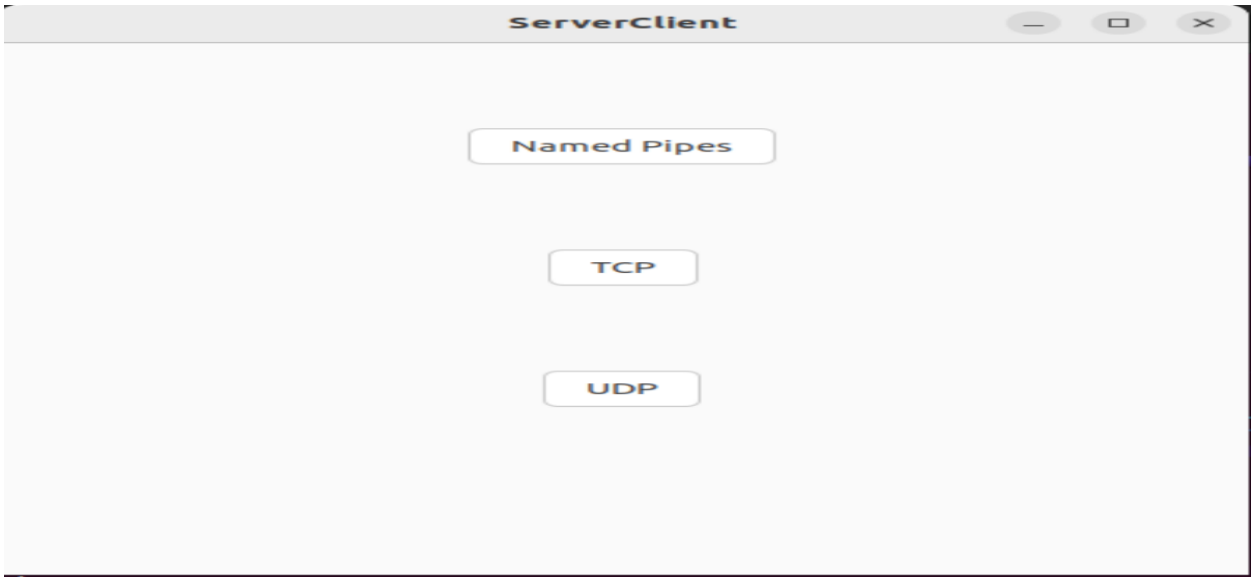

Le bouton "Run Server" permet au serveur de se lancer en terminal et reste ouvert en attendant les clients.



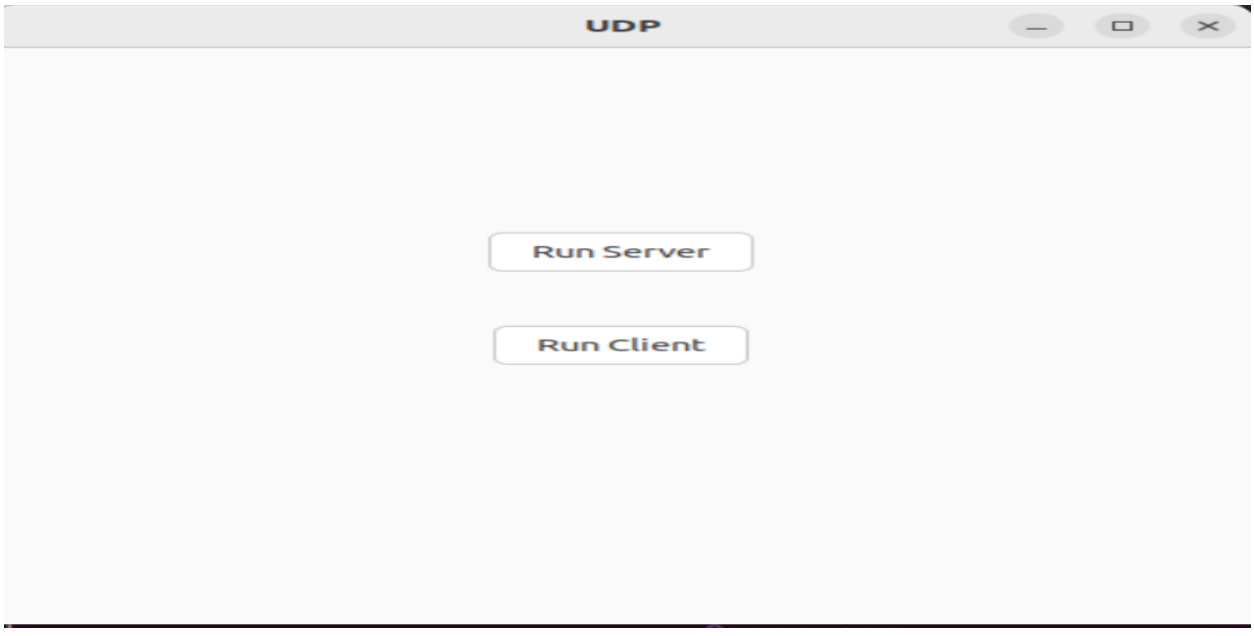

Le bouton “Run Client” génère cette interface qui donne le résultat envoyé en réponse de serveur a la question du client.



La trace d'exécution.

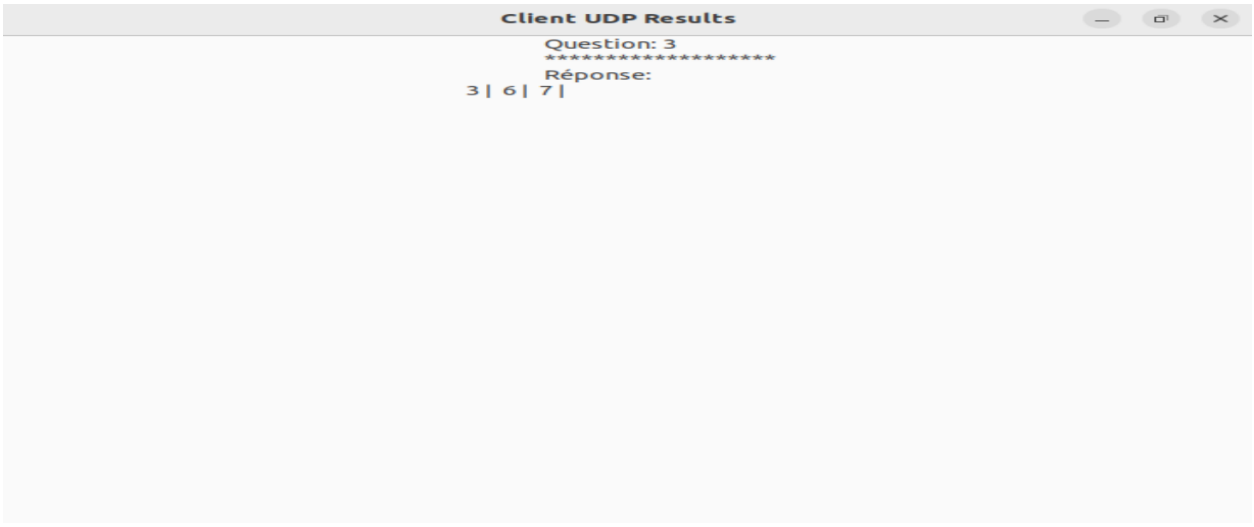



Cette interface se lance lors du choix de l'utilisateur du mode non connecté UDP en cliquant sur le bouton « UDP » .
Les deux boutons visualisé le « Run Server » met le serveur en exécution et l'autre « Run Client » met le client en exécution tout en nottant que le serveur est multi-clients donc on peut lancer plusieurs clients .

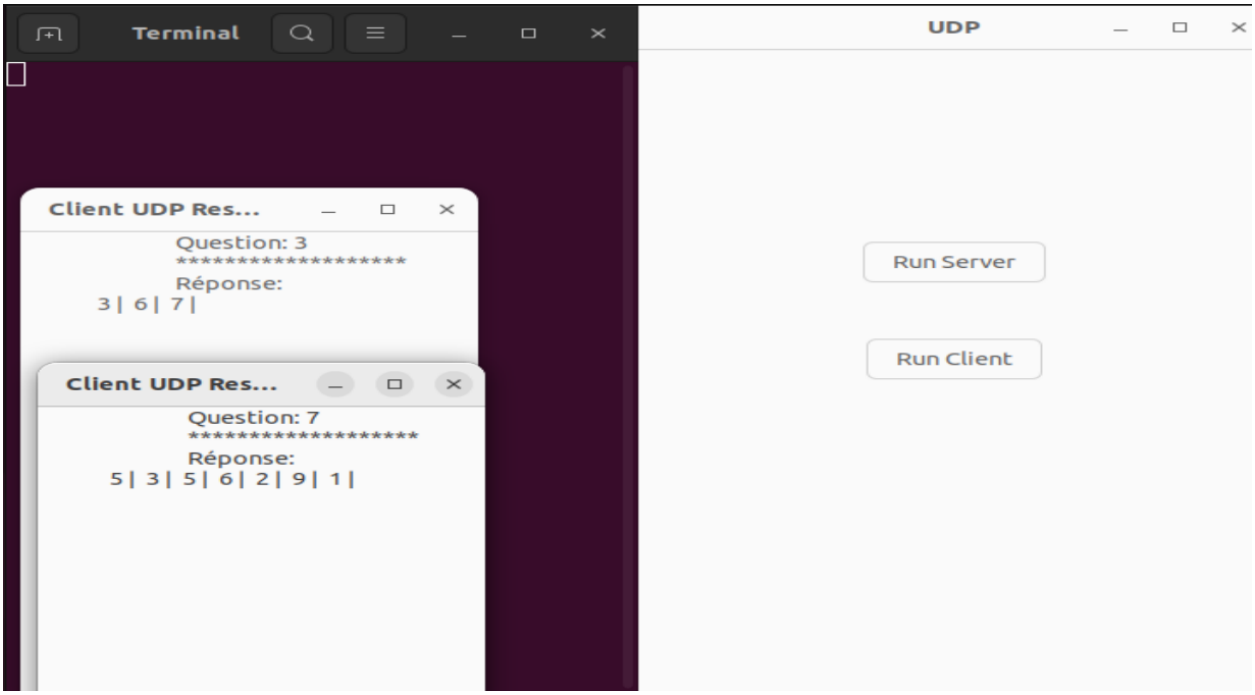



Le bouton "Run Server" permet au serveur de se lancer en terminal et reste ouvert en attendant

les clients.




Le bouton “Run Client” génère cette interface qui donne le résultat envoyé en réponse de serveur a la question du client.




La trace d’exécution.

# Conclusion Générale

Pour recapituler, ce projet consiste à créer un système de communication client/serveur en utilisant les tubes nommés et les sockets en mode connecté en mode non connecté (TCP et UDP).

En termes de développement, on a utilisé le langage C comme langage de programmation pour notre projet.

On a également suivi le modèle en cascade (Waterfall) qui été très convenable avec le projet.

Ce projet était très important, il a permis d'explorer les notions de tubes nommé, sockets, signaux et d'autre concepts