



UNIVERSITÉ DE TUNIS EL MANAR  
INSTITUT SUPÉRIEUR D'INFORMATIQUE



---

# ***Rapport mini-projet (TD) du cours programmation système et réseaux sous UNIX***

---

Préparé par :

**Eya Ben El Kadhi et Eya Bouden (2ING2)**

**Spécialité :** 2ème Année cycle ingénieur de Développement du Logiciel (IDL)

# Table des matières

1. Introduction.....	4
1.1 Contexte du projet.....	4
1.2 Objectifs .....	4
2. Architecture et Conception .....	4
2.1 Partie UDP .....	4
2.2 Partie TCP .....	5
Mono-Mono : .....	5
Multi-Mono : .....	5
Multi-Multi : .....	6
3. Jeu de tests.....	6
3.1 Partie 1 (UDP) .....	6
3.4 Partie 2 (TCP) .....	7
Mono-Mono : .....	7
Multi-Mono : .....	8
Multi-Multi .....	9
5. Difficultés Rencontrées et Solutions .....	12
5.1 Problèmes techniques .....	12
5.2 Solutions apportées.....	12
6. Perspectives d'amélioration .....	13
6.1 Améliorations possibles.....	13
7. Conclusion .....	13

## Table des figures

Figure 1: Compiler et lancer le serveur UDP .....	6
Figure 2: Lancer client_UDP .....	7
Figure 3: Compiler et lancer serveur Mono_Mono .....	7
Figure 4: Lancer client_Mono_Mono .....	8
Figure 5: Compilation avec compiler.sh et affichage du menu_Multi_Mono .....	8
Figure 6: Lancer serveur par le compiler.sh .....	8
Figure 7: Lancer 2 clients en parallèle_Multi_Mono .....	9
Figure 8: Les traitements effectués par le serveur_Multi_Mono .....	9
Figure 9 : Compilation Multiclients-Multiserveurs .....	9
Figure 10: Exécution_Multi_Multi .....	10
Figure 11: Etablir connexion client_Multi_Multi .....	10
Figure 12: Demande du service : Durée de connexion_Multi_Multi .....	10
Figure 13: Demande du service : Contenu des fichiers_Multi_Multi .....	11
Figure 14: Demande du service : Date et heure_Multi_Multi .....	11
Figure 15: Demande du service :Liste des fichiers_Multi_Multi .....	12

# 1. Introduction

## 1.1 Contexte du projet

Le projet s'inscrit dans le cadre du cours de Programmation Système et Réseaux sous UNIX. L'objectif principal est de concevoir et de développer deux applications client/serveur distinctes mettant en œuvre les protocoles de communication UDP et TCP. Ces travaux ont pour but de renforcer la compréhension des mécanismes réseau et de favoriser la maîtrise des concepts avancés de la programmation système sous UNIX.

## 1.2 Objectifs

- Implémenter une communication client/serveur en mode non connecté via le protocole UDP.
- Développement d'une application client/serveur en mode connecté avec le protocole TCP, incluant les configurations monoclient/monoserveur, multiclients/monoserveur, multiclients/multiserveurs, ainsi que l'ajout d'une interface graphique.
- Acquérir une compréhension approfondie des différents paradigmes de programmation réseau sous UNIX.
- Mettre en pratique des compétences liées à la gestion des processus et à la synchronisation.

# 2. Architecture et Conception

## 2.1 Partie UDP

- **ClientUDP:**
  - Création de la socket UDP.
  - Configuration de l'adresse du serveur.
  - Génération d'un nombre aléatoire n.
  - Envoi du nombre n au serveur.
  - Réception des n nombres aléatoires du serveur.
  - Affichage des nombres reçus.
  - Fermeture de la socket.
- **ServeurUDP:**
  - Création de la socket UDP.
  - Configuration de l'adresse du serveur.
  - Liaison de la socket à l'adresse du serveur.
  - Réception du nombre n du client.
  - Génération de n nombres aléatoires.
  - Envoi des n nombres aléatoires au client.
  - Fermeture de la socket.

## 2.2 Partie TCP

### Mono-Mono :

Le serveur gère **un seul client à la fois**.

Une fois qu'un client se déconnecte, le serveur s'arrête.

- **ClientTCP :**

- Création de la socket TCP.
- Configuration de l'adresse du serveur.
- Connexion au serveur.
- Authentification.
- Affichage des services disponibles.
- Demande de service.
- Déconnexion.

- **ServeurTCP:**

- Création de la socket TCP.
- Configuration de l'adresse du serveur.
- Liaison de la socket à l'adresse du serveur.
- Mise en écoute de la socket.
- Acceptation d'une connexion client.
- Gestion du client :
  - ✓ Authentification.
  - ✓ Envoi du menu des services.
  - ✓ Traitement des demandes du client.
  - ✓ Renvoi du menu après chaque service.
- Fermeture de la connexion client.
- Fermeture de la socket du serveur.

### Multi-Mono :

Le serveur gère plusieurs clients simultanément en utilisant des processus enfants. Chaque client est traité indépendamment, et le serveur continue d'accepter de nouvelles connexions même lorsque d'autres clients sont déjà connectés.

Conserver le même client sans modification et le serveur sera adapté pour gérer plusieurs clients simultanément en utilisant la fonction **fork()** pour créer des processus distincts pour chaque connexion client.

#### Serveur :

- Initialisation et écoute des connexions.
- Acceptation des clients et création de processus enfants.
- Authentification des clients.
- Gestion des services et renvoi du menu.
- Fermeture des connexions.

#### Multi-Multi :

Le **proxy** agit comme un intermédiaire central qui gère les connexions client et route les requêtes vers les serveurs de services appropriés. Plusieurs clients peuvent se connecter simultanément au proxy, et chaque client est traité de manière indépendante.

#### Proxy :

- Initialisation et écoute des connexions client.
- Authentification des clients.
- Routage des requêtes vers les serveurs de services.
- Transmission des réponses au client.

#### Serveurs de Services :

- Traitement des requêtes spécifiques (date/heure, liste des fichiers, etc.).
- Envoi des réponses au proxy.

#### Client :

- Connexion au proxy.
- Authentification.
- Envoi des requêtes et réception des réponses

## 3. Jeu de tests

### 3.1 Partie 1 (UDP)

```
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie1$ ./compile_udp.sh
Compilation du client réussie.
Compilation du serveur réussie.
Compilation terminée.
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie1$ ./serveurUDP 12345
Serveur UDP en attente de demandes...
Nombre reçu du client : 25
25 nombres aléatoires envoyés au client.
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie1$
```

Figure 1: Compiler et lancer le serveur UDP

```

eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie1$ ./clientUDP 127.0.0.1 12345
Nombre généré : 25
Nombres reçus du serveur : 25 58 72 60 73 40 98 13 22 87 85 22 45 52 87 87 33 22 1
7 99 7 86 99 2 17
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie1$

```

Figure 2: Lancer client\_UDP

### 3.4 Partie 2 (TCP)

Mono-Mono :

```

eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie2/monoC-mono$ ./compile_tcp.sh
Compilation du client réussie.
Compilation du serveur réussie.
Compilation terminée.
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie2/monoC-mono$ ./serveurTCP 12345
Serveur TCP en attente de connexions...
Client connecté.
Client déconnecté.
Serveur arrêté.
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie2/monoC-mono$

```

Figure 3: Compiler et lancer serveur Mono\_Mono

```

eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie2/monoC-mono$ ./clientTCP 127.0.0.1 12345
Nom d'utilisateur: user
Mot de passe: pass
Authentification réussie.

1. Date et heure
2. Liste des fichiers
3. Contenu d'un fichier
4. Durée de connexion
0. Quitter

Choisissez un service (1-4) ou 0 pour quitter: 1
Réponse du serveur: Tue Jan 21 15:49:41 2025

1. Date et heure
2. Liste des fichiers
3. Contenu d'un fichier
4. Durée de connexion
0. Quitter

Choisissez un service (1-4) ou 0 pour quitter: 2
Réponse du serveur: compile_tcp.sh
serveurTCP.c
serveurTCP
.
clientTCP
clientTCP.c
..
texte.txt

1. Date et heure
2. Liste des fichiers
3. Contenu d'un fichier
4. Durée de connexion
0. Quitter

```

```
Choisissez un service (1-4) ou 0 pour quitter: 4
Réponse du serveur: Durée de connexion: 69.00 secondes

1. Date et heure
2. Liste des fichiers
3. Contenu d'un fichier
4. Durée de connexion
0. Quitter

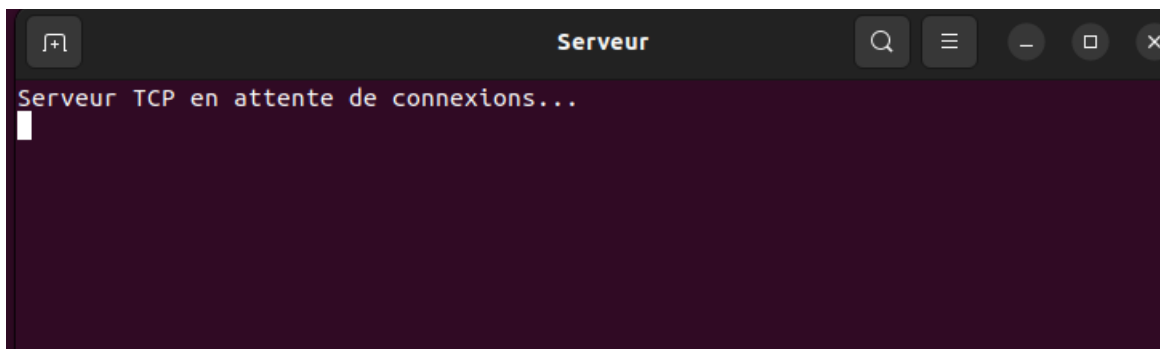
Choisissez un service (1-4) ou 0 pour quitter: 0
Déconnexion...
Client déconnecté. Au revoir !
```

Figure 4: Lancer client\_Mono\_Mono

## Multi-Mono :

```
eya@eya-VirtualBox:~/Desktop/mini-projetUNIX/partie2/multiC-mono$ ./compile.sh
Choisissez une option :
1. Compiler les programmes
2. Lancer le serveur
3. Lancer 2 clients
4. Quitter
Entrez votre choix (1-4) : 1
Compilation du client...
Compilation du client réussie.
Compilation du serveur...
Compilation du serveur réussie.
Choisissez une option :
1. Compiler les programmes
2. Lancer le serveur
3. Lancer 2 clients
4. Quitter
Entrez votre choix (1-4) : 2
Lancement du serveur sur le port 12345 dans un nouveau terminal...
Serveur lancé avec le PID 6965.
Choisissez une option :
1. Compiler les programmes
2. Lancer le serveur
3. Lancer 2 clients
4. Quitter
```

Figure 5: Compilation avec compiler.sh et affichage du menu\_Multi\_Mono



The screenshot shows a terminal window with the title bar 'Serveur'. The window contains the text 'Serveur TCP en attente de connexions...' followed by a cursor. The window has standard Linux window controls (minimize, maximize, close) and a search icon.

Figure 6: Lancer serveur par le compiler.sh



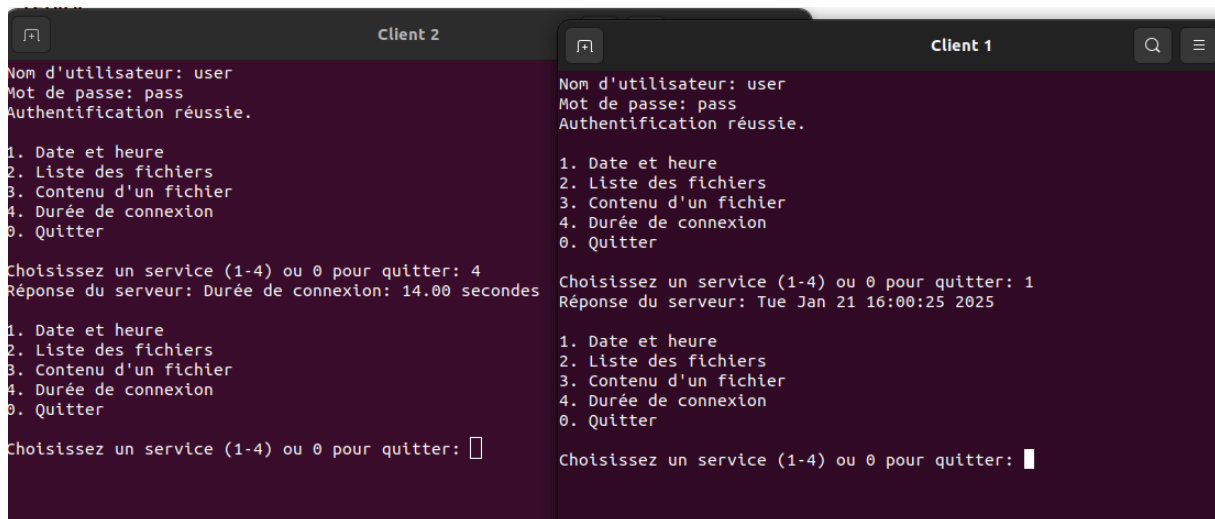


Figure 7: Lancer 2 clients en parallèle\_Multi\_Mono



Figure 8: Les traitements effectués par le serveur\_Multi\_Mono

## Multi-Multi

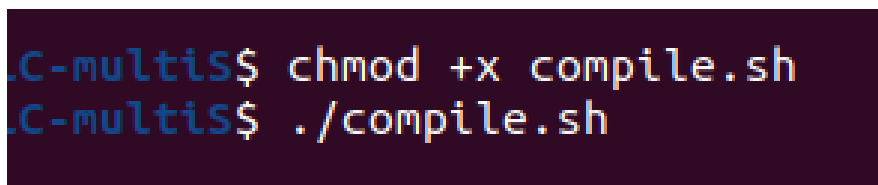


Figure 9 : Compilation Multiclients-Multiserveurs

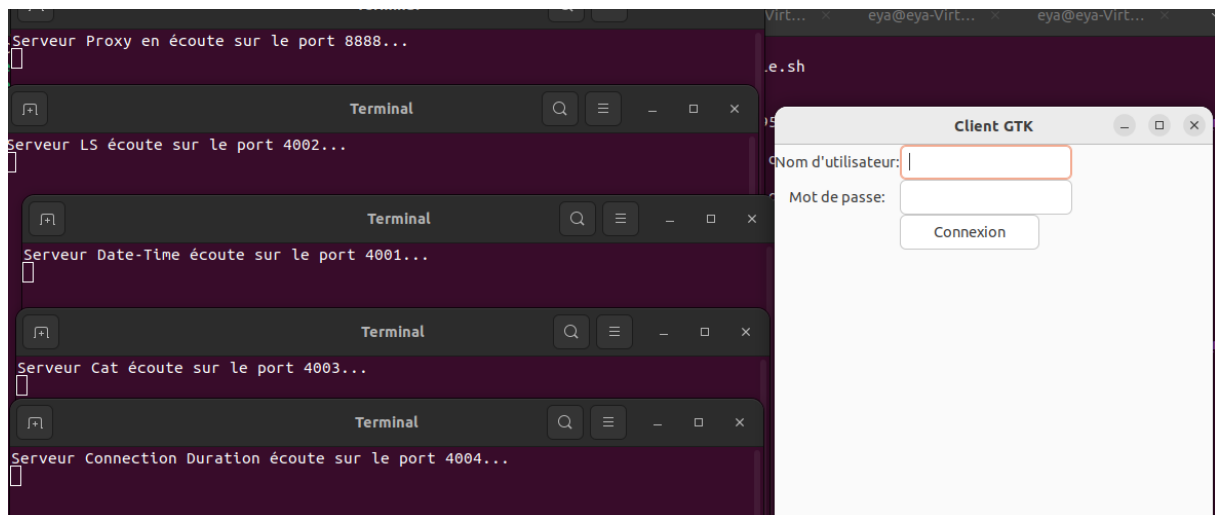


Figure 10: Exécution\_Multi\_Multi

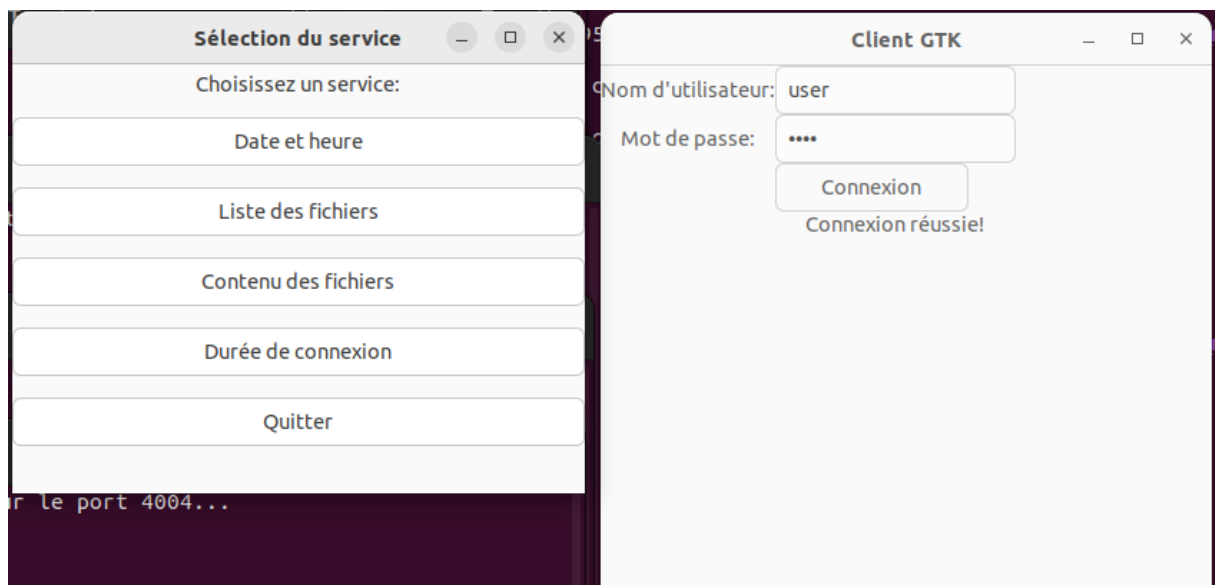


Figure 11: Etablir connexion client\_Multi\_Multi

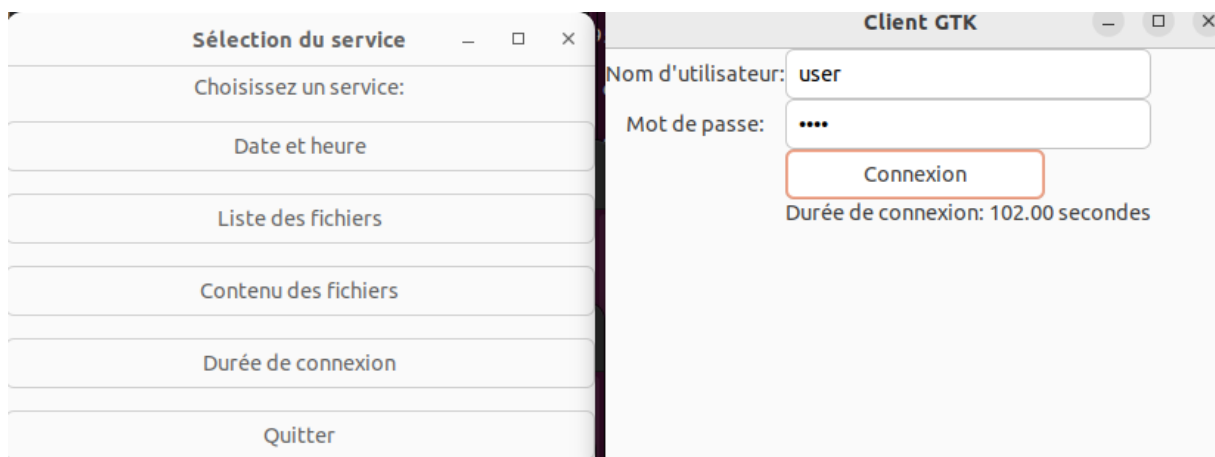


Figure 12: Demande du service : Durée de connexion\_Multi\_Multi

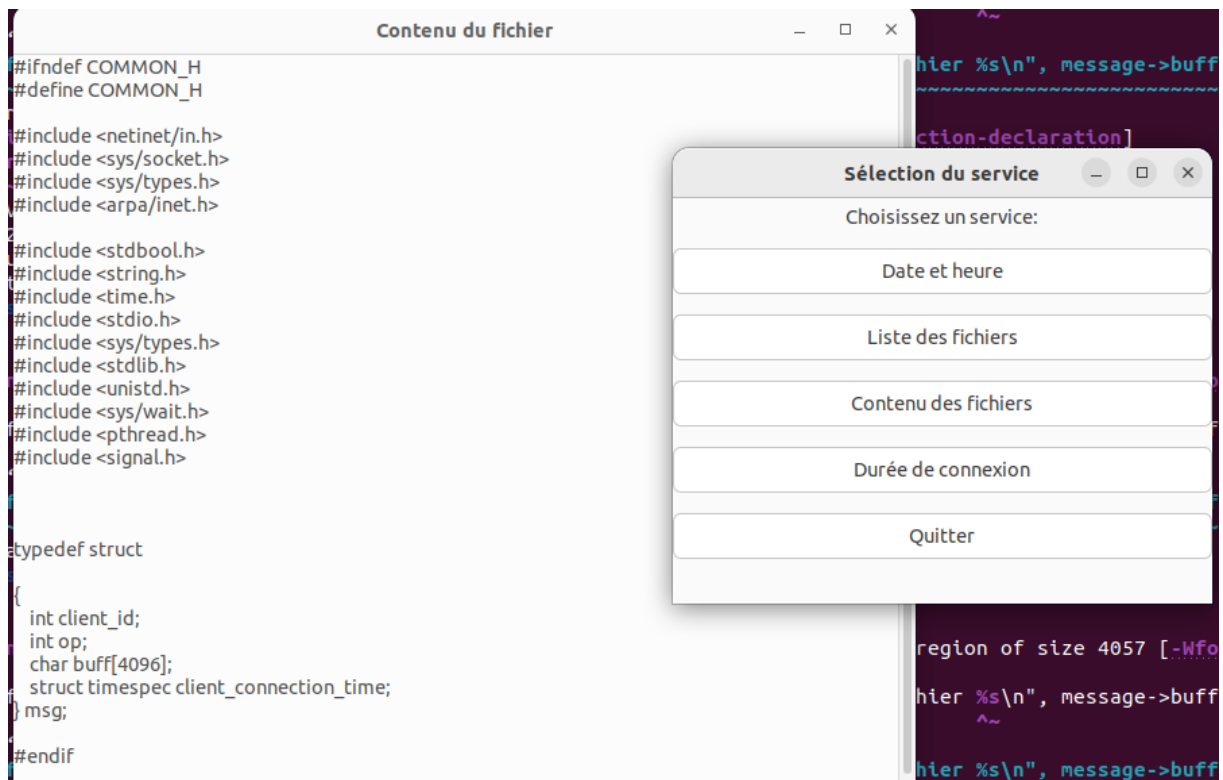


Figure 13: Demande du service : Contenu des fichiers\_Multi\_Multi

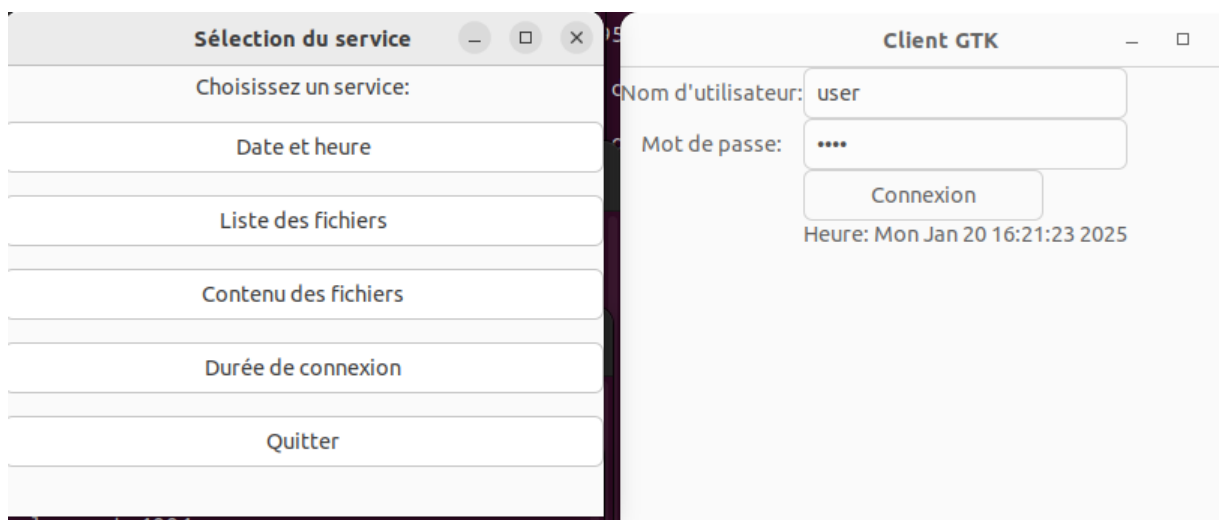


Figure 14: Demande du service : Date et heure\_Multi\_Multi

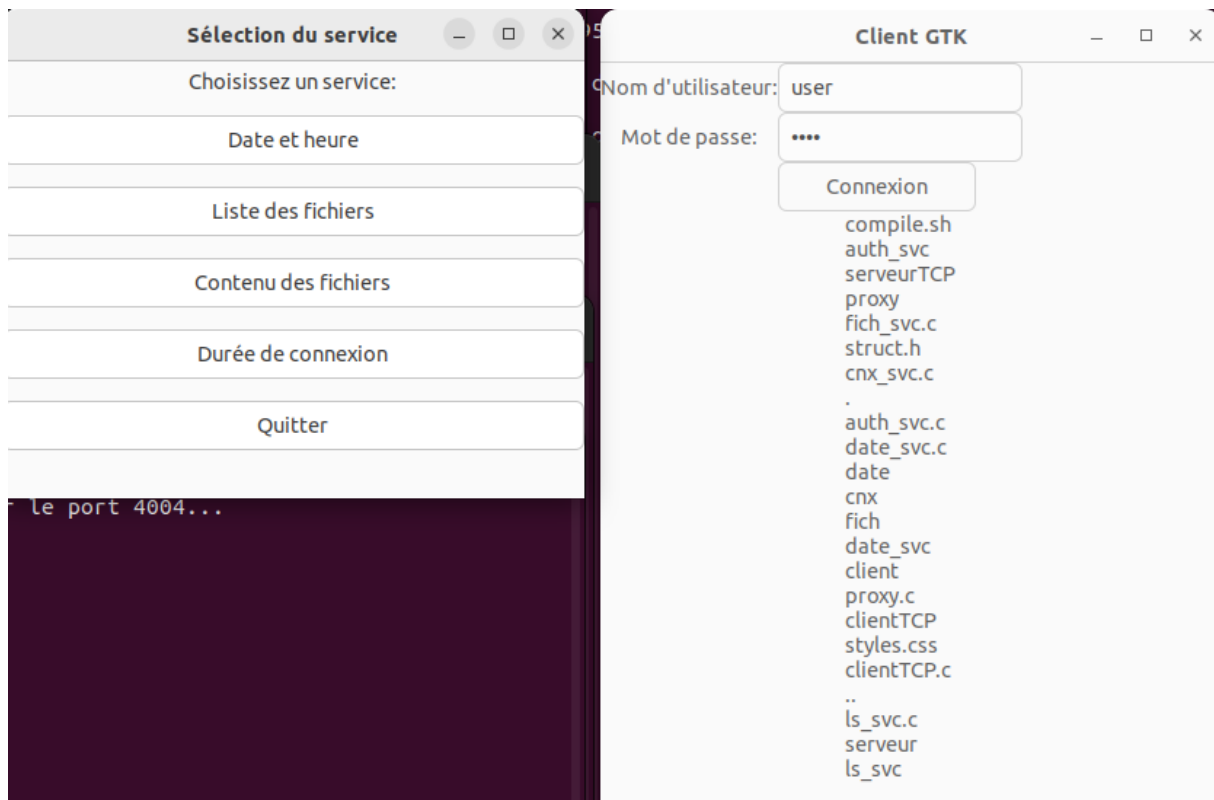


Figure 15: Demande du service :Liste des fichiers\_Multi\_Multi

## 5. Difficultés Rencontrées et Solutions

### 5.1 Problèmes techniques

- **Gestion de la concurrence** : Coordination des processus dans le serveur TCP multi-clients.
- **Configuration de la deuxième partie** : Difficulté à implémenter la gestion multi-serveurs multi-clients, notamment pour détecter le temps de connexion de chaque client.
- **Implémentation des services sur différents ports** : Définir et gérer un mécanisme pour appeler dynamiquement le serveur dédié à un service spécifique.
- **Affichage du contenu d'un fichier** : Lorsqu'un client demandait le contenu d'un fichier, des problèmes tels qu'une mauvaise gestion de la taille du fichier ou des erreurs d'envoi pour des fichiers volumineux rendaient le processus instable.
- **Problèmes d'intégration avec GTK** : Difficultés à intégrer la bibliothèque pour l'interface graphique à cause des dépendances et de la gestion des événements.

### 5.2 Solutions apportées

- **Gestion des connexions multiples** : Implémentation de processus fils grâce à la fonction `fork()` pour chaque nouvelle connexion client.

- **Détection du temps de connexion** : Création d'une structure envoyant un *timestamp* au début de chaque connexion pour calculer la durée exacte.
- **Services sur différents ports** : Mise en place d'un mécanisme où chaque service est assigné à un port dédié, avec un client configuré pour communiquer dynamiquement selon le port requis.
- **Problème d'affichage du contenu d'un fichier** :  
La solution a consisté à :
  1. Vérifier l'existence du fichier et renvoyer un message d'erreur au client en cas d'erreur.
  2. Calculer la taille totale du fichier en utilisant `fseek` et transmettre cette taille au client pour garantir une réception correcte.
  3. Lire le fichier dans un *buffer* dynamique alloué avec `malloc` pour gérer les fichiers de toute taille.
  4. Envoyer le contenu du fichier au client en une seule transmission.
- **Intégration de GTK** : Résolution des dépendances avec des environnements préconfigurés et simplification de l'interface graphique pour éviter les conflits liés à la gestion des threads dans GTK.

## 6. Perspectives d'amélioration

### 6.1 Améliorations possibles

- Ajouter une interface graphique (GUI) plus intuitive et ergonomique.
- Étendre les services offerts par le serveur.
- Renforcer la sécurité des communications (établissement de sessions chiffrées).

## 7. Conclusion

Le projet a permis de concrétiser avec succès les objectifs techniques tout en renforçant les compétences en programmation réseau sous UNIX, offrant une expérience enrichissante et une maîtrise accrue des protocoles UDP et TCP.