SAÉ 3 - R3.05
Simulateur

Contexte

La plateforme de vente en ligne Alizon va confier ses colis à des prestataires de transport. Ces prestataires fournissent un outil permettant :

- La prise en charge du colis
- L'interrogation de la situation du colis dans la chaîne de livraison
- La validation de la livraison

Vous allez concevoir et développer un simulateur de livraisons pour l'un de ces prestataires.

Terminologie

Voici les termes utilisés dans ce document et que vous devez aussi utiliser dans la documentation à produire.

Serveur ou **Service** : le logiciel simulateur qui attend et traite les requêtes.

Client : le logiciel qui se connecte au serveur. Il s'agit du logiciel utilisé par Alizon pour interroger le service. Ne pas confondre avec le client destinataire (voir plus bas).

Protocole : les règles et la grammaire permettant à un client de soumettre des requêtes au serveur.

Grammaire : la syntaxe décrivant comment doivent/peuvent être formatées les requêtes du protocole.

Requête : une action faite sur le serveur par le biais d'un client en utilisant le protocole.

Destinataire : l'acheteur du produit à livrer. Rappel : ne pas confondre avec le "client" du service (voir plus haut).

Attendus

Protocole

Vous devrez concevoir un protocole pour vous identifier et interroger ou faire des actions sur le simulateur.

Vous devrez documenter ce protocole pour en expliquer les règles (sa grammaire, cf TP sur les Sockets)

Langages et outils

Vous devrez fournir le code source ainsi que la documentation nécessaire à la compilation de votre serveur sur une machine Linux (uniquement et exclusivement). Vous devez disposer, sur les machines de l'IUT, de tout ce qui est nécessaire, librairies comprises.

L'utilisation de Git est impérative, sur le Gitlab de l'IUT ou sur un dépôt public (vous devrez alors inviter votre enseignant en tant que Reporter)

Langage C et FIFO

Le simulateur devra être écrit en langage C, mettre en œuvre une programmation de sockets et implémenter des files (FIFO, cf TP Développement efficace).

Options

Le serveur devra être lancé en ligne de commande et disposer d'options (qui sont détaillées au fil de ce document).

Vous pouvez choisir la technique que vous souhaitez pour le passage des options, mais sachez qu'il existe une fonction prévue pour ça : **`getopt()`**.

Note sur l'implémentation

Si vous hébergez votre application Alizon sur les serveurs de l'IUT, compte tenu des contraintes techniques de nos infrastructures, vous ne pourrez pas installer votre service sur le serveur Web, ni effectuer des requêtes depuis vos scripts PHP vers votre service.

Vous testerez donc localement, à l'aide de Telnet ou d'un client développé en C si vous préférez. Vous devrez aussi fournir le code de ce client, si c'est le choix retenu, et fournir une documentation d'utilisation..

Documentation

Vous devrez produire une documentation suffisamment détaillée pour pouvoir compiler et exécuter votre serveur. Ceci doit pouvoir être fait par un développeur, sans nécessiter de connaissances particulières autre que savoir lancer des commandes dans un Terminal.

Protocole

Votre protocole doit permettre de faire les actions suivantes et vous devez proposer une grammaire précise et sans ambiguïté.

Rappel : un protocole est un échange de messages formalisés entre deux pairs, le client et le serveur. Cet échange n'est pas forcément une simple question suivie d'une simple réponse. Il peut (et c'est souvent le cas) être constitué de plusieurs échanges successifs durant la connexion.

A titre d'exemple, un échange téléphonique est dicté par un sorte de protocole tacite :

- Bonjour, que puis-je faire pour vous ?
- Je voudrais savoir où en est la réparation de ma voiture.
- Bien-sûr, donnez-moi son immatriculation.
- C'est AA-999-ZZ.
- Elle est prête mais on a vu que les pneus arrière sont à la limite d'usure.
  On a une promo -20% pour deux pneus, montage et équilibrage offerts ce mois-ci, voulez-vous en profiter ?
- Oui je veux bien.
- C'est noté, vous pourrez venir récupérer votre véhicule à partir de 15:00.
  Bonne journée.
- Merci, à ce soir.

Adaptez votre protocole en fonction des besoins, en vous inspirant de ce principe.

Identification

Pour pouvoir faire une quelconque action, le client doit être authentifié au préalable.

La liste des identifiants possibles et des mots de passe associés (hashés en **`MD5`**) sera placée dans un fichier dont on passera le chemin par une option au lancement.

Prise en charge

Le transporteur a une capacité de prise en charge limitée. Elle est variable et paramétrable par une option passée au lancement du serveur.

La prise en charge n'est plus possible si la file de livraison est pleine. Dès qu'une place se libère dans la file, la prise en charge est de nouveau possible.

Les livraisons terminées et en attente d'accusé de réception seront aussi stockées dans une file d'attente (voir *Cycle du transport*).

Cycle du transport

Le serveur simule le transport avec les étapes suivantes :

- Prise en charge : instantanée
- Transport vers la plateforme régionale : aléatoire, entre 1 et 3 jours
- Transport entre la plateforme et le site local de livraison : 1 jour
- Livraison au destinataire : instantanée, ce qui libère la place dans la file
- Accusé de réception de clôture de cycle : instantané

Une journée théorique sera réduite (accélérée) en un équivalent en minutes. Ce paramètre d'accélération doit être passé sous forme d'une option du service au lancement en ligne de commande.

Attention, la livraison au destinataire libère une place dans la file mais ne doit pas faire perdre la trace du colis tant qu'il n'y a pas eu d'accusé de réception du client. La liste des colis livrés doit être conservée dans une file séparée.

Accusé de réception

Le client interroge le service, à intervalle régulier, pour connaître les livraisons terminées. Le serveur renvoie une liste, éventuellement vide, et le client doit faire savoir qu'il accuse réception de cette liste, ce qui a pour effet de libérer ces livraisons sur le serveur. Sans cet accusé de réception, la liste est gardée intacte et sera donc renvoyée en l'état (complétée d'éventuelles nouvelles livraisons) à l'interrogation suivante par le client.

Service

Votre service est mono-processus et les files sont stockées uniquement en mémoire. On considère donc que chaque lancement se fait à vide (plus aucune livraison en cours).

Votre service doit afficher une aide en ligne par l'utilisation d'une option **`--help`** et, dans ce cas, s'arrêter immédiatement. Inspirez-vous de ce qu'un **`gcc --help`** affiche, par exemple.

Le service écoute sur toutes les interfaces et sur un port que vous devrez passer en option de lancement sur la ligne de commande.

Vous devez produire des logs détaillés, dans un fichier, afin de suivre précisément ce que fait, ce que reçoit, ce que répond, votre service, comment il a été lancé, avec quelles options, etc. Préfixez chaque ligne de log par une date+heure ainsi que l'IP du client[1].

Livrables

Voici la liste des livrables attendus, à déposer sur Moodle :

- Code source C du serveur et autres éventuels outils complémentaires + Documentation pour compiler et pour exécuter en ligne de commande.
- Protocole détaillé (technique)
- Documentation de cas d'utilisation par l'exemple (pratique) : se logguer, soumettre une prise en charge, interroger la situation, accuser réception d'une fin de cycle.
- Une répartition (en %age) du travail individuel par tâche. Au minimum ceci, à détailler/compléter si utile :
    - Conception du protocole
    - Codage et tests
    - Documentation

[1] Voir le paramètre 2 du **`accept()`**