

Enoncé de Réseau & Programmation Réseau V 1.5
(2 gestion , 2 telecom , 2 indus)

Les plateformes de développement

1) Il faut télécharger la librairie de composants de l'école virtuelle, dossier [VANSTAPEL, Herman /Réseau et programmation Reseaux](#)

Nom du fichier	Plateforme0
lib2022GStudent.tar	Oracle linux ou ubuntu

2) Désarchiver dans votre dossier de la plateforme de votre choix

3) **Lib2022GStudent\Step0** contient le code de la première étape du dossier

4) Les exemples du tome T4P se trouvent respectivement dans les dossier ex0X

Les composants de l'application Véhicules

La librairie libSer SHV Les deux dernières lettres sont à remplacer par vos noms respectifs	Cette librairie contient les fonctions partagées entre Admin et le programme Ser
admin	Ce programme permet de créer des véhicules qui sont stockés dans un fichier VehiculesHV . Il permettra aussi de facturer la vente d'automobiles et le résultat sera stocké dans le fichier FactureHV
ser	Ce programme reçoit les requêtes réseaux du programme cli et selon la requête effectue une recherche d'un Véhicule dans le fichier VehiculesHV créé par admin ou une vente qui est stockée alors dans le fichier FactureHV
cli	Ce programme permet à l'utilisateur de consulter les Véhicules disponibles et d'acheter des véhicules. Il les transmet au serveur via des requêtes udp qui lui fournit toujours par réseau les réponses.

Personnalisation du programme

- Le nom de la librairie libSer doit se terminer par vos initiales
- Les structures de données que vous créez (y compris Transaction) doivent se terminer par les initiales.
- Tous les fichiers de données doivent se terminer par vos initiales
- Même principe pour toutes les fonctions ou procédures définies.
- Chaque programme quand il se lance doit afficher le nom et prénom de l'étudiant
- **Vous devrez avoir un champ personnel au niveau de votre application.**

Grille d'évaluation pour les Deuxièmes telecom & Gestion

La librairie réseaux est impérativement udplib.o

Un dossier pour deux personnes .

Chaque équipe reçoit un champ personnel qu'il faut impérativement respecter

L'évaluation est continue pour les trois premiers points de labo.

Les autres points peuvent être présentés au plus tard à l'examen.

Si le programme n'est pas personnalisé comme demandé, champ supplémentaire, c'est d'office 0/20

Le travail dans l'équipe doit être équitable

Je peux demander de faire des modifications

Numéro	Description	A présenter Semaine du (*)
1	Le programme Admin, champ Personnel & Ecrire la fonction recherche	14/11 A présenter avant l'examen
2	Un Client et un seul serveur la requête recherche	28/11 A présenter avant l'examen
3	Un Serveur et plusieurs clients	19/12 A présenter avant l'examen
4A	Le programme admin gérant l'achat et la génération de factures	Au plus tard à l'examen
4B	l'achat et la génération de factures par réseau	Au plus tard à l'examen
5A	L'achat, les clients gèrent les timeout et les doublons, le serveur ne gère pas les doublons	Au plus tard à l'examen
5B	Corrigeons le bug de la recherche	Au plus tard à l'examen
6,7	L'achat, les clients gèrent les timeout et les doublons, le serveur traite les doublons , Le calcul du crc	Au plus tard à l'examen
	Remise de votre dossier Lib avec toutes vos étapes par mail à herman.vanstapel@hepl.be. <u>Pénalité si recopie ou non remise du dossier</u>	Au plus tard avant l'examen

Grille d'évaluation pour les Deuxièmes indus

Pour les conditions voir telecom.

Numéro	Description	A présenter Semaine du (*)
1	Le programme Admin, champ Personnel & Ecrire la fonction recherche	17 / 4
2	Un Client et un seul serveur la requête recherche	1 / 5
3	Un Serveur et plusieurs clients	15 / 5
4A	Le programme admin gérant l'achat et la génération de factures	Au plus tard à l'examen
4B	l'achat et la génération de factures par réseau	Au plus tard à l'examen
5A	L'achat, les clients gèrent les timeout et les doublons, le serveur ne gère pas les doublons	Au plus tard à l'examen
5B	Corrigeons le bug de la recherche	Au plus tard à l'examen
6,7	L'achat, les clients gèrent les timeout et les doublons, le serveur traite les doublons , Le calcul du crc	Au plus tard à l'examen
	Remise de votre dossier Lib avec toutes vos étapes par mail à herman.vanstapel@hepl.be. <u>Pénalité si recopie ou non remise du dossier</u>	Au plus tard avant l'examen

Grille d'évaluation pour les Deuxièmes Gestion

La liste des champs personnels à rajouter à séances

Numéro	Nom du champ	Description	Valeurs
1	Cylindrée		1200, 1500, 2000
2	Puissance		136 , 192 (des chevaux)
3	Carrosserie		SuV, monospace, Break, berline
4	Couleur		Jaune, Vert , Bleu , Noir
5	Transmission		Avant , Arrière , 4 X 4
6	Motorisation		Essence, diesel, Electrique , Hybride
7	Portes		3,4,5
8	Boite		Manuelle, automatique
9	Jantes		15,16, 17
10	Kilometrage		20000 , 30000 , 100000 , 200000

1)Le programme Admin, champ Personnel & Ecrire la fonction recherche

documentation

T4P , annexe les opérations sur les fichiers.

Un admin de base à compléter est fourni dans lib2022GEtud\step0

Copier le contenu de ce dossier dans lib2022GEtud\step1

Le makefile

```
# cphex\makefile

LIBS=
all:    admin  LibSerHV.o

LibSerHV.o:    LibSerHV.c    LibSerHV.h    data.h
               echo "compilation de LibSerHV"
               gcc -c LibSerHV.c

admin: data.h  admin.c LibSerHV.o
        echo "Compilation de admin"
        gcc -o admin    admin.c LibSerHV.o
```

Rappelons que LibSer doit être renommé avec vos initiales

Compilation du programme

```
# cphex\makefile

LIBS=
all:    admin  LibSerHV.o

LibSerHV.o:    LibSerHV.c    LibSerHV.h    data.h
               echo "compilation de LibSerHV"
               gcc -c LibSerHV.c

admin: data.h  admin.c LibSerHV.o
        echo "Compilation de admin"
        gcc -o admin    admin.c LibSerHV.o
```

Modifiez data.h pour votre champ perso et la saisie et l'affichage

```
#ifndef DATAH

#else

#define DATAH
#include <time.h>
// 2022 , interdit de modifier les champs existants
```

```
struct VehiculeHV {
    int Supprime ;
    int Reference ;
    char Constructeur[30] ;
    char Modele[30] ;
    int Quantite ;

};

struct FactureHV
{
    int NumeroFacturation ;
    char Acheteur[40] ;
    time_t DateFacturation ;
    int Quantite;
    int Reference ;
};

#endif
```

En première étape, il vous est demandé de rajouter un champ à VehiculeHV que je donnerais
Vous modifier la saisie des Véhicules et leur affichage

Modifier la fonction a propos dans LibSerHV.h

Faire l’affichage de vos noms & prénoms

Ecrire la fonction recherche

La fonction suivante est à écrire, Le prototype est à type indicatif, elle sera placée dans libSerHV.c

```
int RechercheHV(char* NomFichier,int Reference ,struct VehiculeHV
*UnRecord) ;
```

int Recherche retourne 1 si la recherche a réussi.

Char *NomFichier désigne chaque fois le nom du fichier qui contient les Véhicules,

Reference : le numéro du véhicule à chercher

struct VehiculeHV* pointera vers le Record qui contiendra le résultat de la recherche

Je vous conseille vivement de lire le code de la fonction existante SaiSieRecord pour se rappeler la syntaxe d’un pointeur vers une structure. Regarder aussi CreationAjoutFichier ;

Fonctionnement illustré

```
vanstapel@vanstap2:~/lib2022G/Step1$ ./admin 2>logT
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit
```

```

-----
>>2
Ref Constructeur      Modele      Quantite
1 Dacia              AAAA          1
2 Dacia              BBBB          4
3 EEEE              RRRR          3
4 FFFF              SSSSS         10
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit
-----
>>4
Saisie Reference:3
Ref Constructeur      Modele      Quantite
3 EEEE              RRRR          3      (**)
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit
-----
>>

```

En **(**)** vous devez rajouter votre propre champ personnel

Le log d'admin

```

vanstapel@vanstap2:~/lib2022G/Step1$ cat logT
Ouverture reussie
Record lu 1 et Position actuelle dans le fichier 72
Record lu 1 et Position actuelle dans le fichier 144
Record lu 1 et Position actuelle dans le fichier 216
Record lu 1 et Position actuelle dans le fichier 288
RechercheHV>
Ouverture reussie VehiculesHV
Reference lue 1 et Position actuelle dans le fichier 72
Reference lue 2 et Position actuelle dans le fichier 144
RechercheHV<

```

2) Un Client et un seul serveur la requête recherche

documentation

T4P EX02 : Un serveur et un seul client : structure de donnée

Préalables

Copier le contenu du dossier step1 dans le dossier step2. Y copier aussi le contenu de EX02

Le makefile

Il est conseillé d'utiliser le makefile suivant :

```
LibSerHV.o:    LibSerHV.c    LibSerHV.h    data.h
    echo "compilation de LibSerHV"
    cc -c LibSerHV.c

admin: data.h  admin.c LibSerHV.o
    echo "Compilation de admin"
    cc -o admin      admin.c LibSerHV.o

udplib.o:      ../udplib/udplib.h      ../udplib/udplib.c
    echo "Compilation de udplib.o"
    cc -c ../udplib/udplib.c

cli:   cli.c   requetehv.h   data.h   udplib.o
    echo "Compilation de client"
    cc -o cli cli.c      udplib.o      $(LIBS)

ser:   ser.c   requetehv.h   data.h   udplib.o LibSerHV.o
    echo "Compilation de serveur"
    cc -o ser ser.c      udplib.o LibSerHV.o      $(LIBS)
```

LibSer est à compléter avec vos initiales ainsi que requete

En cas de problèmes de compilation, effacer les exécutables cli et ser et effacer tous les .o

Dans l'exemple 02 l'include structure.h doit être remplacé par un include requetehv.h dont voici le contenu :

La structure requetehv.h

```
enum TypeRequete {
    Question = 1 ,
    Achat = 2 ,
    Livraison= 3 ,
    OK = 4,
    Fail = 5
};

struct RequeteHV
{
    enum TypeRequete Type ;
    int Numero ; // Contient le numéro de la requete
    int NumeroFacture ;
    time_t Date ;
    int Reference ; // la référence du film
    int Quantite ;
    int Prix ;
    char Constructeur[30] ;
    char Modele[30] ;
    char NomClient[80] ;
};

#include <stdio.h>

void AfficheRequeteHV(FILE *fp, struct RequeteHV R )
{
    fprintf(fp,">TypeRequete %d \n", R.Type ) ;
    fprintf(fp," Reference %d \n", R.Numero ) ;
    fprintf(fp," NumeroFacture %d \n", R.NumeroFacture ) ;
    fprintf(fp," Date %ld \n", R.Date ) ;
    fprintf(fp," Reference %d \n", R.Reference ) ;
    fprintf(fp," Places %d \n", R.Quantite ) ;
    fprintf(fp," Prix %d \n", R.Prix ) ;
    fprintf(fp,"Constructeur %s \n", R.Constructeur ) ;
    fprintf(fp,"Modele %s \n", R.Modele ) ;
    fprintf(fp," Client %s \n\n", R.NomClient ) ;
    return ;
}
```

La structure requête est utilisées dans les échanges réseaux entre le programme **cli** et le programme **ser**.
La structure et le fichier doivent être renommées avec vos initiales.

L'affichage du client

Le client demande la saisie d'une référence et communique celle-ci au serveur qui répond avec le véhicule correspondant.

```
vanstapel@vanstap2:~/lib2022G/Step2$ ./cli 127.0.0.1 1400 127.0.0.1 1300 2>LogCli
CreateSockets 3
Saisie reference vehicule:1
bytes ecrits 176
bytes lus 176
Constructeur , Modele:Dacia AAAA (**)
```

(**) champ perso

```
vanstapel@vanstap2:~/lib2022G/Step2$ cat LogCli
port 1400
>TypeRequete 4 // On affiche la requête avant envoi
Reference 0
NumeroFacture 0
Date 0
Reference 1
Places 0
Prix 0
Constructeur Dacia
Modele AAAA
Client
```

L'affichage du serveur

```
vanstapel@vanstap2:~/lib2022G/Step2$ ./ser 127.0.0.1 1300 2>LogSer
Ceci est le serveur
CreateSockets : 3
bytes Lus 176 Reference:1
bytes ecrits 176
```

Le serveur utilise la fonction recherche incluse dans LibSerHV.c

```
vanstapel@vanstap2:~/lib2022G/Step2$ cat LogSer
port 1300
>TypeRequete 1           // On affiche la requête à la réception
Reference 0
NumeroFacture 0
Date 0
Reference 1
Places 0
Prix 0
Constructeur
Modele
Client

RechercheHV>
Ouverture reussie VehiculesHV
RechercheHV<
res :1 Reference:Dacia AAAA (**)
```

() Vous devrez rajouter comme information le champ qui vous est propre.**

3) Un Serveur et plusieurs clients

Documentation

Consulter Ex03 : Un serveur et plusieurs clients Structure de donnée du tome T4P

Préalables

Copier le contenu du dossier step2 dans le dossier step3.

La structure à adopter au niveau du code du serveur

Un switch case est recommandé à cette étape pour pouvoir intégrer les étapes suivantes.

```
rc = ReceiveDatagram( Desc,&UneRequeteR ,tm, &sor ) ;
if ( rc == -1 )
    perror("ReceiveDatagram") ;
else
    printf("bytes Lus %d Reference:%d\n",rc,UneRequeteR.Reference ) ;
    AfficheRequeteHV(stderr, UneRequeteR ) ;
    // Le traitement doit être séparé des requêtes réseaux */
    printf("Received packet from %s:%d\n", inet_ntoa(sor.sin_addr), ntohs(sor.sin_port));
    fprintf(stderr,"Received packet from %s:%d\n", inet_ntoa(sor.sin_addr), ntohs(sor.sin_port));
    switch(UneRequeteR.Type )
    {
    case Question:
        res = RechercheHV("VehiculesHV",UneRequeteR.Reference ,&UnRecord) ;
        fprintf(stderr,"res :%d Reference:%s %s\n",res,UnRecord.Marque, UnRecord.Modele) ;
        /* reponse avec psor qui contient toujours l'adresse du dernier client */

        memset(&UneRequeteE,0, sizeof(struct RequeteHV)) ;
        if ( res==1)
        {
            UneRequeteE.Reference = UneRequeteR.Reference ;
            strncpy(UnRequeteE.Marque,UnRecord.Marque,sizeof(UnRequeteE.Marque)) ;
            strncpy(UnRequeteE.Modele,UnRecord.Modele,sizeof(UnRequeteE.Modele)) ;
            // A completer
```

Le fonctionnement du serveur

```
vanstapel@vanstap2:~/lib2022G/Step3$ ./ser 127.0.0.1 1300 2>logser
```

Ceci est le serveur

CreateSockets : 3

bytes Lus 176 Reference:1

Received packet from 127.0.0.1:1400

bytes ecrits 176

bytes Lus 176 Reference:2

Received packet from 127.0.0.1:1500

bytes ecrits 176

() Vous devrez rajouter comme information le champ qui vous est propre.**

Fonctionnement illustré des deux clients

Le client intègre maintenant un menu. Il y'a deux clients maintenant.

Chaque client demande la référence du véhicule à chercher et envoie l'identifiant au serveur par réseau.

Le serveur affiche l'ip et le port du client qui s'est connecté.

Le serveur répond en fournissant la marque, le modèle du véhicule, la quantité restante et le champ perso.

```
vanstapel@vanstap2:~/lib2022G/Step3$ ./cli 127.0.0.1 1500 127.0.0.1 1300 2>log1500
```

CreateSockets 3

1) Demander une reference

3) Quitter

Choix :1

Reference :2

bytes écrits 176

bytes lus 176

Pour 2 Constructeur, Modele Dacia BBBBB Quantite: 4 **(**)**

1) Demander une reference

3) Quitter

Choix :

```
vanstapel@vanstap2:~/lib2022G/Step3$ ./cli 127.0.0.1 1400 127.0.0.1 1300 2>logcli  
CreateSockets 3
```

```
-----  
1) Demander une reference  
3) Quitter
```

```
-----  
Choix :1  
Reference :1  
bytes écrits 176  
bytes lus 176  
Pour 1 Constructeur , Modele Dacia AAAA Quantite: 1 (**)
```

```
-----  
1) Demander une reference  
3) Quitter
```

```
-----  
Choix :
```

(**) champ personnel

4A) Le programme admin gérant l'achat et la génération de factures

Préalables

Copier le contenu du dossier step3B dans le dossier step04a

Le fonctionnement illustré

Il faut modifier le programme admin pour ajouter une option **Achat**. J'achète une certaine quantité de véhicules. Le programme diminue La quantité des véhicules commandées et ajoute une facture au nom de l'acheteur dans le fichier facture

```
vanstapel@vanstap2:~/lib2022G/Step4A$ ./admin 2>logCli
```

```
-----2022-----
```

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

```
-----
```

```
>>2
```

Ref	Constructeur	Modele	Quantite	
1	Dacia	AAAA	1	(**)
2	Dacia	BBBBB	2	(**)
3	EEEE	RRRR	2	(**)

```
-----2022-----
```

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

```
-----
```

```
>>5
```

```
NomClient :Spiderman 2022
```

```
Reference :3
```

```
Quantite:1
```

```
Trouve EEEE , RRRR Quantite 2
```

```
Mise à jour du Fichier FactureHV réussie
```

```
-----2022-----
```

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

```
-----
>>6
1  TOTO HER96                1  2  16:00
2  TOTOHER 96                1  3  16:04
3  Spiderman 2022            1  3  18:11
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit
-----
>>
```

()** champ perso

On notera que 4 est le numéro de la facture (on avait déjà créé des factures)

Exemple de prototypes de fonctions à rajouter à **LibSerHV.h**

int **ReservationHV**(char* NomFichier,int Reference ,int Quantite) ;

int **FacturationHV**(char NomFichier[80], char NomClient[60], time_t Date,int Quantite,int Reference) ;

4B) l'achat et la génération de factures par réseau

Préalables

Copier le contenu du dossier step4A dans le dossier step4B

On demande maintenant d'intégrer la fonction achat au menu client et d'intégrer les fonctions Réservation & Facturation au Serveur

Vérifier le stock avec admin

```
vanstapel@vanstap2:~/lib2022G/Step4B$ ./admin 2>logAdmin
```

```
-----2022-----
```

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

```
-----
```

```
>>2
```

Ref	Constructeur	Modele	Quantite
1	Dacia	AAAA	1
2	Dacia	BBBBB	1
3	EEEE	RRRR	2
4	Ford	Fiesta	4

L'affichage du client

```
vanstapel@vanstap2:~/lib2022G/Step4B$ ./cli 127.0.0.1 1400 127.0.0.1 1300 2>logCli
```

```
CreateSockets 3
```

```
-----
```

- 1) Demander une reference
- 2) Acheter un Véhicule
- 3) Quitter

```
-----
```

```
Choix :2
```

```
NomClient :VANSTAP2022
```

```
Reference :4
```

```
Quantite:1
```

```
bytes ecrits 176
```

```
bytes lus 176
```

```
Achat Reussi Facture : 5
```

Le fonctionnement illustré du serveur

```
vanstapel@vanstap2:~/lib2022G/Step4B$ ./ser 127.0.0.1 1300 2>logSer
```

Ceci est le serveur

CreateSockets : 3

bytes Lus 176 Reference:4

Received packet from 127.0.0.1:1400

Trouve Ford , Fiesta Quantite 4

Mise à jour du nombre de la quantite réussi

bytes écrits 176

(**) Ne pas oublier champ personnel

Vérification du résultat avec admin

```
vanstapel@vanstap2:~/lib2022G/Step4B$ ./admin 2>logAdmin
```

-----2022-----

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

>>2

Ref	Constructeur	Modele	Quantite
1	Dacia	AAAA	1
2	Dacia	BBBBB	1
3	EEEE	RRRR	2
4	Ford	Fiesta	3

-----2022-----

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

>>6

1	TOTO	HER96	1	2	16:00
2	TOTO	HER 96	1	3	16:04
3	VANSTAP	98	1	2	2:00
4	VANSTAP	98	1	4	14:40
5	VANSTAP2022		1	4	15:48

5A) L'achat, les clients gèrent les timeout et les doublons, le serveur ne gère pas les doublons

Documentation

Ex07 du tome4P

Préalables

Copier le contenu du dossier step4B dans le dossier step5A

Analyse

Les transactions doivent maintenant être numérotées de manière à détecter les doublons au niveau du client. le client chaque fois qu'il transmet une demande d'achat, démarre un timer. Si timeout , le client retransmet la demande telle quelle sans incrémenter le numéro de transaction.


Au niveau du serveur via le ctrl z, on mettra le serveur en pause d'environ 30 secondes. Le serveur ne répond pas pendant ce temps aux requêtes du client. **Quand le serveur se réveille il devra répondre à toutes les requêtes du client.**

Le client vérifie que le paquet reçu correspond bien au numéro de transaction envoyé. **Si ce n'est pas le cas , il affiche doublon** et se remet en attente du bon numéro de transaction

Utilisons admin

```
vanstapel@vanstap2:~/lib2022G/Step5AB$ ./admin
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit
-----
>>2
Ouverture reussie
Ref Constructeur      Modele      Quantite
Record lu 1 et Position actuelle dans le fichier 72
1 Citroen              AX          10          // Voir Exercice 5B
Record lu 1 et Position actuelle dans le fichier 144
2 Citroen              BX          11
Record lu 1 et Position actuelle dans le fichier 216
3 Citroen              CX          8
```

Le fonctionnement illustré du client

```
vanstapel@vanstap2:~/lib2022G/Step5AB$ ./cli 127.0.0.1 1600 127.0.0.1 1300
port 1600
CreateSockets 3
-----
1) Demander une reference
2) Acheter un Véhicule
3) Quitter
-----
Choix :2
NomClient :Mercredi
Reference :3
Quantite:1
>TypeRequete 2
Reference 0
NumeroFacture -1633635008
Reference 3
Places 1
Prix -1633363480
Constructeur 
Modele
Client Mercredi
13:57
bytes écrits 176
error sur receive:: Interrupted system call
rc -1 errno:4
rc -1 errno:4
bytes écrits 176
bytes lus 176
>TypeRequete 4
Reference 0
NumeroFacture 17
Reference 0
Places 0
Prix 0
Constructeur
Modele
Client
2:00
Achat Reussi Facture : 17
```

Attention si vous faites des achats en premier, la fonction demander une référence ne donnera plus de résultats corrects car elle peut être perturbée par les doublons générés par L'option acheter

Le fonctionnement illustré du serveur

vanstapel@vanstap2:~/lib2022G/Step5AB\$./ser 127.0.0.1 1300

Ceci est le serveur

port 1300

CreateSockets : 3

^Zlongjumped from interrupt CTRL Z 20

Demarrage du sleep

Fin du sleep

bytes Lus 176 Reference:3

>TypeRequete 2


Reference 0

NumeroFacture -1633635008

Reference 3

Places 1

Prix -1633363480

Constructeur 

Modele

Client Mercredi

13:57

Received packet from 127.0.0.1:1600

Received packet from 127.0.0.1:1600

ReservationHV>

Ouverture reussie VehiculesHV

Record lu 1 et Position actuelle dans le fichier 72

Record lu 2 et Position actuelle dans le fichier 144

Trouve Citroen , CX Quantite 8

Trouve Citroen , CX Quantite 8

Record Ecrits 1

ReservationHV<

FacturationHV>

Ouverture reussie de FactureHV

FacturationHV<

Mise à jour du nombre de la quantite réussi

bytes écrits 176

bytes Lus 176 Reference:3

>TypeRequete 2

Reference 0

NumeroFacture -1633635008

Reference 3

Places 1

Prix -1633363480

Constructeur 

Modele

Client Mercredi

13:57

Received packet from 127.0.0.1:1600

Received packet from 127.0.0.1:1600

ReservationHV>

Ouverture reussie VehiculesHV
Record lu 1 et Position actuelle dans le fichier 72
Record lu 2 et Position actuelle dans le fichier 144
Trouve Citroen , CX Quantite 7
Trouve Citroen , CX Quantite 7
Record Ecrits 1
ReservationHV<
FacturationHV>
Ouverture reussie de FactureHV
FacturationHV<
Mise à jour du nombre de la quantite réussi
bytes écrits 176

() Champ Perso**

Constater la diminution des places et les factures créées avec admin

-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit

>>2
Ouverture reussie
Ref Constructeur Modele Quantite
Record lu 1 et Position actuelle dans le fichier 72
1 Citroen AX 10
Record lu 1 et Position actuelle dans le fichier 144
2 Citroen BX 11
Record lu 1 et Position actuelle dans le fichier 216
3 Citroen CX 6
-----2022-----
1) Ajout
2) Vehicule
4) Recherche
5) Achat
6) Factures
7) A propos
8) exit

>>6
Ouverture reussie
Record lu 1 et Position actuelle dans le fichier 64
1 TOTO 1 2 2:00
Record lu 1 et Position actuelle dans le fichier 128
// les enregistrements 2 à 13 ne sont pas montrés pour que le listing ne soit pas trop long

14 vendredi 2	1 3 16:03
Record lu 1 et Position actuelle dans le fichier 960	
15 vendredi 2	1 3 16:03
Record lu 1 et Position actuelle dans le fichier 1024	
16 vendredi 2	1 3 16:03
Record lu 1 et Position actuelle dans le fichier 1088	
17 Mercredi	1 3 13:57
Record lu 1 et Position actuelle dans le fichier 1152	
18 Mercredi	1 3 13:57

Le serveur à cette étape ne gère pas les doublons. Il répond à toute requête.
Ce sera corrigé plus tard à l'étape 6

5B) Corrigeons le bug de la recherche

Documentation

Ex07 du tome4P

Préalables

Copier le contenu du dossier step5A dans le dossier step5B

Après le point 5A, si vous faites une recherche après avoir fait des achats ou des timeouts se sont produits, vous verrez que cela ne fonctionne pas **car les doublons des achats sont lus avant le résultat de la recherche**

Modifier le code de la recherche pour que maintenant Les transactions de recherche soient également numérotées. Intégrer le timeout n'est pas nécessaire mais il faut intégrer au minimum le code de la gestion des doublons

Refaire les manipulations du Point 5A , un achat sur la **référence 3** et puis faire une recherche sur la **référence 1**

Vous devriez obtenir quelque chose comme ce qui suit :

```
1) Demander une reference
2) Acheter un Véhicule
3) Quitter
-----
Choix :1
Reference :1
bytes écrits 176
>TypeRequete 1
Reference 1
NumeroFacture 0
Reference 1
Places 0
Prix 0
Constructeur
Modele
Client
  2:00
bytes lus 176
doublon 0 !!!!!
>TypeRequete 1
Reference 1
NumeroFacture 0
Reference 1
Places 0
Prix 0
Constructeur
Modele
Client
  2:00
bytes lus 176
Pour 1 Constructeur , Modele Citroen AX Quantite: 10 (**)
```

(**) Champ perso

6) L'achat, les clients gèrent les timeout et les doublons, le serveur traite les doublons

Documentation

Ex07 du tome4P

Préalables

Copier le contenu du dossier step5 dans le dossier step6

Analyse

Quand les timers se déclenchent, le serveur répond toujours au client car il ne sait savoir si les paquets sont arrivés. Le problème est que le serveur ne vérifie pas si une facture a déjà été établie pour le client pour l'achat. Donc si on a trois doublons pour une quantité commandée de un véhicule, on va retirer quatre fois la place alors que le client n'en a demandé qu'un véhicule.

Pour corriger ce problème, le client doit fournir un renseignement supplémentaire qui est la date d'achat générée par le système. Le serveur fait une recherche sur le nom de client et la date.

- Si aucune correspondance n'est trouvée, Le serveur génère une nouvelle facture et réduit le nombre les stock de véhicules.
- Si on trouve une correspondance, le serveur retourne au client le numéro de facture déjà généré et ne change pas la quantité

NE PAS OUBLIER DE FAIRE L'AFFICHAGE DU CHAMP PERSO.

Vérifier le stock de départ dans admin

```
vanstapel@vanstap2:~/lib2022G/Step6$ ./admin 2>logadmin
```

```
-----2022-----
```

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

```
-----
```

```
>>2
```

Ref	Constructeur	Modele	Quantite
1	Citroen	AX	10
2	Citroen	BX	11
3	Citroen	CX	14

L'affichage du client

```
vanstapel@vanstap2:~/lib2022G/Step6$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
-----
1) Demander une reference
2) Acheter un Véhicule
3) Quitter
-----
Choix :2
NomClient :Vanstap jeudi
Reference :3
Quantite:1
>TypeRequete 2
Reference 0
NumeroFacture -1176402624
Reference 3
Places 1
Prix -1176131096
Constructeur @
Modele
Client Vanstap jeudi
11:47
bytes écrits 176
error sur receive:: Interrupted system call      TIMEOUT
rc -1 errno:4
rc -1 errno:4
bytes écrits 176
error sur receive:: Interrupted system call      TIMEOUT
rc -1 errno:4
rc -1 errno:4
bytes écrits 176
error sur receive:: Interrupted system call      TIMEOUT
rc -1 errno:4
rc -1 errno:4
bytes écrits 176
>TypeRequete 4
Reference 0
NumeroFacture 11
Reference 0
Places 0
Prix 0
Constructeur
Modele
Client
2:00
bytes lus 176
```

Achat Reussi Facture : 11

L'affichage du serveur

vanstapel@vanstap2:~/lib2022G/Step6\$./ser 127.0.0.1 1300

Ceci est le serveur

port 1300

CreateSockets : 3

^Zlongjumped from interrupt CTRL Z 20

Demarrage du sleep

Fin du sleep

bytes Lus 176 Reference:3

>TypeRequete 2

Reference 0

NumeroFacture -1176402624

Reference 3

Places 1

Prix -1176131096

Constructeur @

Modele

Client Vanstap jeudi

11:47

Received packet from 127.0.0.1:1400

Received packet from 127.0.0.1:1400

RechercheFactureDateHV>

Ouverture reussie FactureHV

RechercheFactureDateHV<

Requete inexistante

ReservationHV>

Ouverture reussie VehiculesHV

Record lu 1 et Position actuelle dans le fichier 72

Record lu 2 et Position actuelle dans le fichier 144

Trouve Citroen , CX Quantite 14

Trouve Citroen , CX Quantite 14

Record Ecrits 1

ReservationHV<

FacturationHV>

Ouverture reussie de FactureHV

FacturationHV<

Mise à jour du nombre de la quantite réussi

bytes écrits 176

bytes Lus 176 Reference:3

>TypeRequete 2

Reference 0

NumeroFacture -1176402624

Reference 3

Places 1

Prix -1176131096

Constructeur @

Modele

Client Vanstap jeudi

11:47

Received packet from 127.0.0.1:1400

Received packet from 127.0.0.1:1400

RechercheFactureDateHV>

Ouverture reussie FactureHV

RechercheFactureDateHV<

Doublon

bytes écrits 176

bytes Lus 176 Reference:3

>TypeRequete 2

Reference 0

NumeroFacture -1176402624

Reference 3

Places 1

Prix -1176131096

Constructeur @

Modele

Client Vanstap jeudi

11:47

Received packet from 127.0.0.1:1400

Received packet from 127.0.0.1:1400

RechercheFactureDateHV>

Ouverture reussie FactureHV

RechercheFactureDateHV<

Doublon

bytes écrits 176

bytes Lus 176 Reference:3

>TypeRequete 2

Reference 0

NumeroFacture -1176402624

Reference 3

Places 1

Prix -1176131096

Constructeur @

Modele

Client Vanstap jeudi

11:47

Received packet from 127.0.0.1:1400

Received packet from 127.0.0.1:1400

RechercheFactureDateHV>

Ouverture reussie FactureHV

RechercheFactureDateHV<

Doublon

bytes écrits 176

() champ personnel**

Vérifier le stock et la création de facture via le programme admin **

-----2022-----

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

>>2

Ref	Constructeur	Modele	Quantite
1	Citroen	AX	10 (**)
2	Citroen	BX	11 (**)
3	Citroen	CX	13 (**)

-----2022-----

- 1) Ajout
- 2) Vehicule
- 4) Recherche
- 5) Achat
- 6) Factures
- 7) A propos
- 8) exit

>>6

1	TOTO	1	2	2:00
2	TOTO	1	2	2:00
3	TOTO	1	2	2:00
4	TOTO	1	2	2:00
5	AAAAA 2409	1	1	2:00
6	111	1	1	2:00
7	1	1	1	2:00
8	1	1	1	2:00
9	1	1	1	2:00
10	VANVAN	1	3	13:55
11	Vanstap jeudi	1	3	11:47

(**) champ perso

7) Communication entre une machine Linux ubuntu et Linux Oracle

Documentation

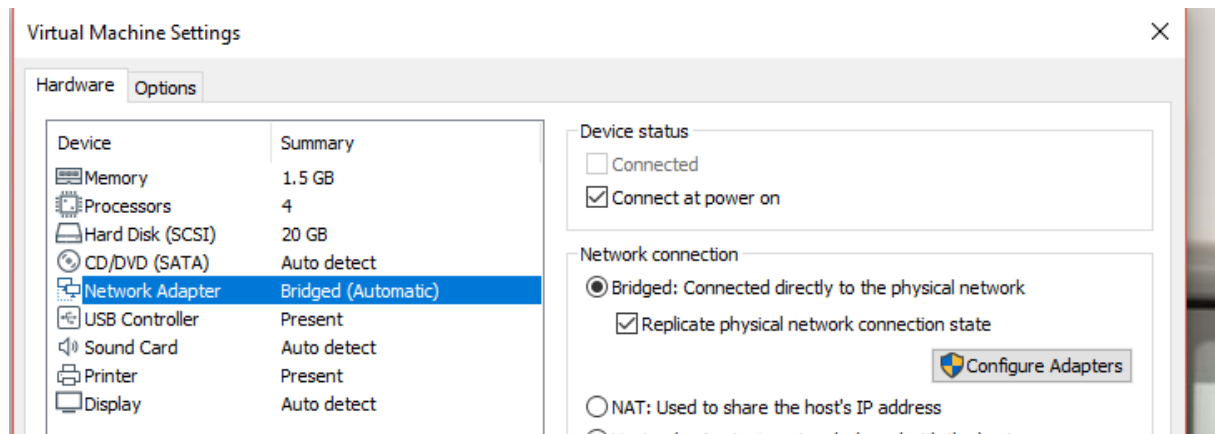
Voir Tome4P, exemple ex08.

Analyse

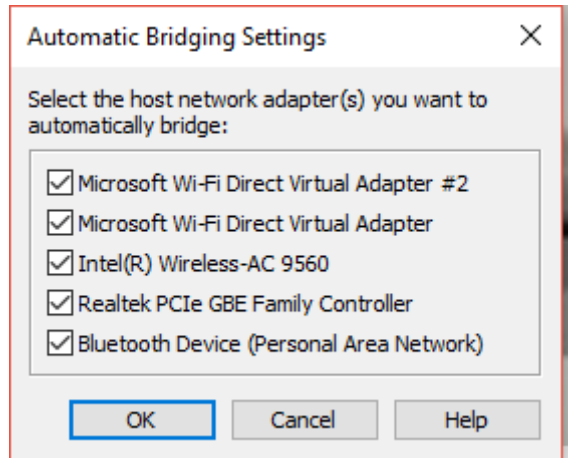
On vous demande de démarrer un serveur sur machine Oracle et de faire la recherche à partir d'un client ubuntu. Les entiers doivent être convertis avec les fonctions hton et ntoh.

Configurer les deux machines en Bridged Replicate

S'assurer que la machine virtuelle est arrêtée



Cliquer sur le bouton **Configure Adapters**



Décocher les adaptateurs qui posent problème . Typiquement , virtual box et wiresharck peuvent engendrer des problèmes

Obtenir l'ip de la machine ubuntu

```
vanstapel@vanstap2:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.29 netmask 255.255.255.0 broadcast 192.168.1.255
```

Faire un ping pour vérifier que la connexion fonctionne bien

```
vanstapel@vanstap2:~$ ping 192.168.1.30
```

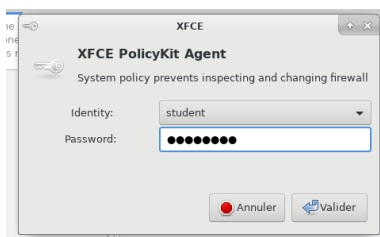
```
PING 192.168.1.30 (192.168.1.30) 56(84) bytes of data.  
64 bytes from 192.168.1.30: icmp_seq=1 ttl=64 time=1.57 ms  
64 bytes from 192.168.1.30: icmp_seq=2 ttl=64 time=1.25 ms
```

Le serveur sous oracle

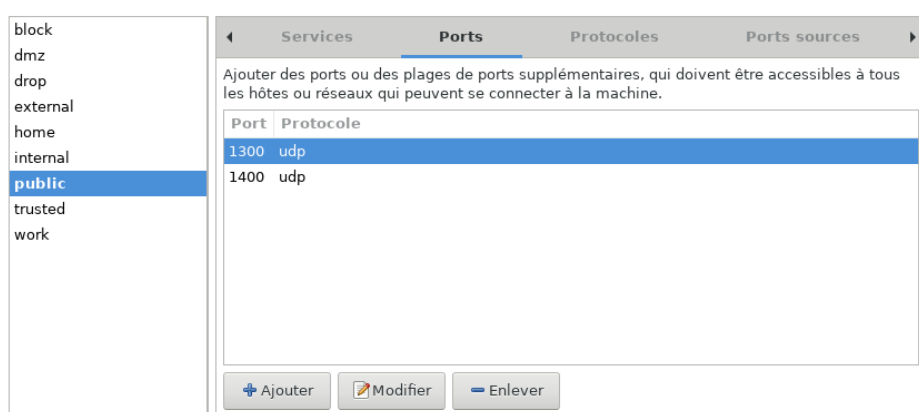
```
[student@zeus ex03]$ ifconfig  
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.30 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 2a02:a03f:a087:1500:bf65:b5b4:86ae:4c46 prefixlen 64 scopeid 0x0
```

```
[student@zeus Step3]$ ./ser 192.168.1.30 1300  
Ceci est le serveur  
port 1300  
CreateSockets : 3
```

Il faut modifier les règles du firewall
Aller dans le menu **administration / pare-feu**



Mot de passe student1



Dans public , onglet ports Rajouter 1300 udp et 1400 udp

Le client sous ubuntu

```
vanstapel@vanstap2:~/lib202021/Step3$ ./cli 192.168.1.29 1400 192.168.1.30 1300 2>log
CreateSockets 3
-----
1) Demander une reference
3) Quitter
-----
Choix :1
Reference :1
bytes écrits 168
bytes lus 168
Pour 1 Marque , Modele Volvo AAAA Quantite: 1
```

Cela ne fonctionne pas

Vérifiez bien les ips et les ports

Eventuellement redémarrer les machines virtuelles respectives. Rester trop longtemps en pause peut faire perdre l'ip acquise par dhcp.

Les points 8 à 10 ne sont pas à présenter dans le cadre du cours de réseau & programmation

8) Le serveur multiport (Pour les gestion, indus dispensé ; telecom pour administration & sec)

Documentation

Ex05 : un serveur multiclents du tome 4P

Analyse

Le serveur écoute chaque requête sur un seul port.

Dans la version multi port, le serveur écoute les clients sur différents ports par exemple 1301, 1302 ou 1304. Pour tester, il suffit de connecter un client sur le port 1301 , un autre sur le 1302, un dernier sur le 1304.

On demande d'adapter la fonction recherche du point 3 au minimum au multiport.

9) La recherche sous forme de thread (indus dispensé ; telecom pour administration & sec)

On demande d'écrire une fonction ThreadRecherche dans **ser.c** qui appellera la fonction recherche et répondre au client.

La syntaxe de cette fonction est la suivante

```
void *ThreadRecherche ( void* Param )
```

Il est impératif de passer une copie des paramètres car sans copie. Imaginons qu'un second thread soit lancé avant l'achèvement du premier thread. Le second thread modifierait les paramètres du premier thread qui répondrait plus à son client . C'est pour cela qu'il est impératif de faire une copie. On déclare en premier une structure ST , a compléter par vos initiales.

```
struct STXX {                               /* XX vos initiales */
    int DescPublic ;
    struct sockaddr_in psorPublic ; /* r = remote */
    struct Requete UneRequeteR ;
};
```

Voici maintenant comment procéder dans le programme principal ser.c

```
struct STXX *pST ;
    pST = malloc( sizeof( struct ST )) ; // chaque thread doit avoir une copie unique des paramètres
    pST->DescPublic = Desc ;
    pST->psorPublic = psor ;
    pST->UneRequeteR = UneRequeteR ;
    .....
    Lancement du thread
```

Pour récupérer les paramètres dans ThreadRecherche, Voici comment faire.

```
void *ThreadRecherche ( void* Param )
{
    struct STXX *pST ;
    struct Requete UneRequeteE ;
    int DescPublic ;
    struct sockaddr_in psorPublic ;
    struct Requete UneRequeteR ;

    int res,rc ;
    fprintf(stderr,"Demarrage du Thread & attente section critique \n") ;
    pST = ( struct STXX * ) Param ;
    DescPublic = pST->DescPublic ;
    psorPublic = pST->psorPublic ;
    UneRequeteR = pST->UneRequeteR ;
```

ThreadRecherche doit être lancé en mode detach, voici la syntaxe

```
rc=pthread_attr_init(&attr);
rc=pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
```

9) La fonction Facturation sous forme de thread en utilisant une section critique (indus dispensé ; telecom pour administration & sec)

Il faut maintenant créer la fonction suivante

```
void *ThreadFacturation ( void* Param )
```

qui sera appelé toujours en mode detach et le passage des paramètres se fera toujours comme expliqué au point 8.

La modification des fichier doit être impérativement protégée par une section critique.

Il est impératif de déclarer le mutex en global dans la fonction ser pour qu'il fonctionne correctement

```
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER; // Toujours déclaré public
```

Pour montrer l'effet du mutex , un affichage sera fait avant l'entrée dans la section critique, mutex_lock et après l'entrée, ainsi qu'un sleep pour permettre de lancer un second thread alors que le premier n'est pas achevé

```
fprintf(stderr,"Demarrage du Thread & attente section critique \n") ;  
pthread_mutex_lock( &mutex1 );  
fprintf(stderr,"Entrée Section critique\n") ;  
sleep(20) ;
```