

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
lib2020Etud.tar	Sun ou Linux

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

3) Lib2020Etud\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 Voitures

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 voitures qui sont stockés dans un fichier Voitures HV . Il permettra aussi de facturer la vente de voitures 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'une voiture dans le fichier VoituresHV créé par admin ou une vente qui est stockée alors dans le fichier FactureHV
cli	Ce programme permet à l'utilisateur de consulter les voitures disponibles et de les acheter. 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

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	18/11 A présenter avant l'examen
2	Un Client et un seul serveur la requête recherche	2/12 A présenter avant l'examen
3	Un Serveur et plusieurs clients	16/12 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 Communication entre machine Linux et SUN	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.

Grille d'évaluation pour les Deuxièmes Gestion

La liste des champs personnels à rajouter à Voiture

Numéro	Nom du champ	Description	Valeurs
1	Cylindrée		1000 , 1500, 2000
2	Puissance	ch	100, 136, 192
3	Boite de Vitesse		Manuelle, Automatique, Palettes
4	Couleur Carrosserie		Jaune, Rouge, Vert
5	Portes		3, 4, 5
6	Freins		Disque, Tambours , Disque ventiléq
7	Finition		Intens , Life , Zen , finitions disponibles pour la captur
8	Carburant		E5 , E10, B7, B10
9	CO2	Gr / KM	95 , 100, 145
10	Type	Type Carrosserie	Berline, break, monospace ,citadines , crossover

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 lib2020Etud\step0

Copier le contenu de ce dossier dans lib2020Etud\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
```

Pour exécuter : taper make. Si des problèmes de compilation se produisent, effacer les fichier admin et libSerHV.o

Rappelons que LibSer doit être renommé avec vos initiales

Si sun

```
LIBS=-lsocket -lnsl
```

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

```
#ifndef DATAH
#else
#define DATAH

struct VoitureHV {
    int Reference ;
    char Marque[40] ;
    char Modele[40] ;
    int Nombre ;
};

struct FactureHV
{
    int NumeroFacturation ;
    char NomClient[40] ;
    int DateFacturation ;
    int Places ;
    int Reference ;
};

#endif
```

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

Modifier la fonction a propos

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 RechercheVoitureHV(char* NomFichier,int Reference , struct
VoitureHV *UnRecord) ;
```

int Recherche retourne 1 si la recherche a réussi.

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

Reference : le numéro de la voiture à chercher

struct VoitureHV* 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 de admin

- 1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit

2

Ouverture reussie

Ref	Marque	Modele	Nombre
1	Ford	Fiesta	3
2	Ford	Focus	3
3	Ford	Mondeo	3
4	Ford	Mustang	2

- 1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit

3

Saisie Reference:3

Ouverture reussie VoituresHV

Reference lue 1 et Position actuelle dans le fichier 88

Reference lue 2 et Position actuelle dans le fichier 176

Ref	Marque	Modele	Nombre	
3	Ford	Mondeo	3	(**)

Les messages en *italique* sont des messages de log. Ils permettent de détecter plus facilement les erreurs en cas de problèmes

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

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 :

```
# Step2\makefile

LIBS=

all:    admin  cli      ser      udplib.o 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

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

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

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

LibSer est à compléter avec vos initiales

Si vous travaillez avec SUN ; complétez la variable LIBS comme ci-dessous

```
LIBS=-lsocket -lnsl
```

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

La structure structure.h

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

struct RequeteHV
{
    enum TypeRequete Type ;
    int Numero ; // Contient le numéro de la requete
    int NumeroFacture ;
    int Date ;    // Utilisé à partir de l'étape 6
    int Reference ; // Reference produit
    int Nombre ;
    int Prix ;
    char Marque[80] ;
    char Modele[80] ;
    char NomClient[80] ;
};
```

La structure requête est utilisées dans les échanges réseaux entre le programme **cli** et le programme **ser**.

Le menu du client

Note en italique, vous avez les informations de log

```
vanstap@ubuntu:~/lib20200630/Step2$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
Saisie Reference Vehicule:1
Envoi de 268 bytes

bytes:268
Marque:Ford Modele:Fiesta (**)
Saisie Reference Vehicule:
```

L'affichage du serveur

```
vanstap@ubuntu:~/lib20200630/Step2$ ./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
CreateSockets : 3
bytes:268 Reference:1
Ouverture reussie VoituresHV (**)
res :1 :Ford Fiesta
bytes:268
```

Le serveur utilise la fonction recherche incluse dans lib2018.c

() 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.

Fonctionnement illustré des deux clients

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

Chaque client demande la référence fi véhicule à chercher et envoie l'identifiant au serveur par réseau qui répond en fournissant la marque , le modèle et le nombre restant et le champ perso.

```
vanstap@ubuntu:~/lib20200630/Step3$ ./cli 127.0.0.1 1400 127.0.0.1 1300
```

```
port 1400
```

```
CreateSockets 3
```

```
-----  
1) Demander une reference
```

```
3) Quitter  
-----
```

```
Choix :1
```

```
Reference :1
```

```
Envoi de 268 bytes
```

```
bytes:268
```

```
Pour 1 Ford,Fiesta Nombre : 3 (**)
```

```
-----  
1) Demander une reference
```

```
3) Quitter  
-----
```

```
Choix :
```

```
vanstap@ubuntu:~/lib20200630/Step3$ ./cli 127.0.0.1 1500 127.0.0.1 1300
```

```
port 1500
```

```
CreateSockets 3
```

```
-----  
1) Demander une reference
```

```
3) Quitter  
-----
```

```
Choix :1
```

```
Reference :2
```

```
Envoi de 268 bytes
```

```
bytes:268
```

```
Pour 2 Ford,Focus Nombre : 3 (**)
```

```
-----  
1) Demander une reference
```

```
3) Quitter
```

Choix :

(**) champ personnel

La structure à adopter au niveau du code du serveur

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

```
switch(UneRequeteR.Type )
{
case Consultation:
    fprintf(stderr,"A rechercher %d \n", UneRequeteR.Reference ) ;
    res = RechercheVoitureHV("VoituresHV",UneRequeteR.Reference ,&UnRecord) ;
    fprintf(stderr,"res :%d %s,%s Places %d\n",res,UnRecord.Marque,UnRecord.Modele,
UnRecord.Nombre ) ;
    /* reponse avec psor qui contient toujours l'adresse du dernier client */
    strncpy(UneRequeteE.Marque,UnRecord.Marque,sizeof(UneRequeteE.Marque)) ;
    strncpy(UneRequeteE.Modele,UnRecord.Modele,sizeof(UneRequeteE.Modele)) ;
    UneRequeteE.Nombre = UnRecord.Nombre ;
    UneRequeteE.Numero = UneRequeteR.Numero ;
    UneRequeteE.Reference = UneRequeteR.Reference ;
    if (res)
        UneRequeteE.Type = OK ;
    else
        UneRequeteE.Type = Fail ;
    rc = SendDatagram(Desc,&UneRequeteE ,sizeof(struct RequeteHV) ,&sor ) ;
    if ( rc == -1 )
        perror("SendDatagram:") ;
    else
        fprintf(stderr,"bytes:%d\n",rc ) ;
    break ;
case Achat:
```

Le fonctionnement du serveur

```
vanstap@ubuntu:~/lib20200630/Step3$ ./ser 127.0.0.1 1300
```

Ceci est le serveur

port 1300

CreateSockets : 3

bytes:268 Type:1 Numero:0

Received packet from 127.0.0.1:1400

A rechercher 1

Ouverture reussie VoituresHV

res :1 Ford,Fiesta Places 3

bytes:268

bytes:268 Type:1 Numero:0

Received packet from 127.0.0.1:1500

A rechercher 2

Ouverture reussie VoituresHV

Reference lue 1 et Position actuelle dans le fichier 88

res :1 Ford,Focus Places 3

bytes:268

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

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**. Je précise la référence et le nombre de voitures souhaitées. **Le programme modifie le nombre de véhicules en stock en retirant le nombre de voitures commandées** et ajoute une facture au nom de l'acheteur dans le fichier facture

```
vanstap@ubuntu:~/lib20200630/Step4A$ ./admin
-----
1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----
3
Saisie Reference:1
Ouverture reussie VoituresHV
Ref Marque      Modele      Nombre
1 Ford          Fiesta      3    (**)
-----
1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----
4
NomClient :TOTO2000
Reference :1
Nombre:1
Ouverture reussie VoituresHV
Trouve Ford,Fiesta
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour du Fichier FactureHV réussie
```

()** est le champ personnel.

On va maintenant vérifier que le nombre a été décrémenté et que la facture a été ajoutée dans le fichier facture

```
-----
1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----

3
Saisie Reference:1
Ouverture reussie VoituresHV
Ref Marque      Modele      Nombre
1  Ford          Fiesta      2
-----

1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----

5
Ouverture reussie
Record lu 1 et Position actuelle dans le fichier 56
1  Vanstap      2  2  0
Record lu 1 et Position actuelle dans le fichier 112
2  TOTO        1  2  0
Record lu 1 et Position actuelle dans le fichier 168
3  TOTO2000    1  1  0
```

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

L'implémentation

Pour réserver la quantité commandée ; dans LibSerHV ajouter la ligne suivante :

```
int ReservationVoitureHV(char* NomFichier,int Reference ,int Nombre )
```

Attention la fonction retournera 0 si les Places ne sont pas disponible, sinon elle décrémente pour la référence de la voiture du nombre commandé.

Dans LibSerHV; il faut ajouter la ligne suivante pour générer la facture

```
int FacturationHV(char NomFichier[80], char NomClient[60], int Date,int Nombre,int Reference)
```

Pour le moment le champ date, vous le mettez à zéro.

Pour générer les numéros de facture , vous pouvez utiliser la formule suivante :

UneFacture.NumeroFacturation = ftell(sortie) / sizeof(struct Facture) + 1 ;

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 Reservation &

Fonctionnement du client

```
vanstap@ubuntu:~/lib20200630/Step4B$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
```

```
-----
1) Demander une reference
2) Acheter une Voiture
3) Quitter
-----
```

```
Choix :1
Reference :1
Envoi de 268 bytes
```

```
bytes:268
Pour 1 Ford,Fiesta Nombre: 3      (**)
```

```
-----
1) Demander une reference
2) Acheter une Voiture
3) Quitter
-----
```

```
Choix :2
NomClient :Freddy
Reference :1
Nombre:1
Envoi de 268 bytes
```

```
bytes:268
Achat Reussi Facture : 2
```

```
-----
1) Demander une reference
2) Acheter une Voiture
3) Quitter
-----
```

```
Choix :
```

() est votre champ personnel**

Le fonctionnement illustré du serveur

```
vanstap@ubuntu:~/lib20200630/Step4B$ ./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
CreateSockets : 3
bytes:268 Type:1 Numero:0
Received packet from 127.0.0.1:1400
A rechercher 1
Ouverture reussie VoituresHV
res :1 Ford,Fiesta 3
bytes:268
bytes:268 Type:2 Numero:1
Received packet from 127.0.0.1:1400
Ouverture reussie VoituresHV
Trouve Ford,Fiesta
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour du nombre réussi
bytes:268
```

(**) Ne pas oublier champ personnel

Vérification du résultat avec admin

```
vanstap@ubuntu:~/lib20200630/Step4B$ ./admin
```

- ```

1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit

```

```
3
```

```
Saisie Reference:1
```

```
Ouverture reussie VoituresHV
```

| Ref | Marque | Modele | Nombre |
|-----|--------|--------|--------|
| 1   | Ford   | Fiesta | 2      |

- ```
-----  
1) Ajout  
2) Voitures  
3) Recherche  
4) Achat  
5) Factures  
6) A propos  
7) exit  
-----
```

```
5
```

```
Ouverture reussie
```

```
Record lu 1 et Position actuelle dans le fichier 56
```

1	Vanstap	1	3	0
---	---------	---	---	---

```
Record lu 1 et Position actuelle dans le fichier 112
```

2	Freddy	1	1	0
---	--------	---	---	---

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

Le fonctionnement illustré du client

```
vanstap@ubuntu:~/lib20200630/Step5AB$ ./cli 127.0.0.1 1400 127.0.0.1 1300
```

port 1400

CreateSockets 3

-
- 1) Demander une reference
 - 2) Acheter une Voiture
 - 3) Quitter
-

Choix :1

Reference :6

Envoi de 268 bytes

bytes:268

Pour 6 Ford,GT 50 Nombre: 6 **(**)**

-
- 1) Demander une reference
 - 2) Acheter une Voiture
 - 3) Quitter
-

Choix :2

NomClient :spiderman du lundi

Reference :6

Nombre:1

Envoi de 268 bytes

error sur receive:: Interrupted system call

rc -1 errno:4

Envoi de 268 bytes

error sur receive:: Interrupted system call

rc -1 errno:4

Envoi de 268 bytes

bytes:268

Achat Reussi Facture : 8

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

```
vanstap@ubuntu:~/lib20200630/Step5AB$ ./ser 127.0.0.1 1300
```

Ceci est le serveur

port 1300

CreateSockets : 3

bytes reçus:268 Type:1 Numero:0

Received packet from 127.0.0.1:1400

A rechercher 6

Ouverture reussie VoituresHV

res :1 Ford,GT 50 Nombre 6 **(**)**

bytes:268

^Zlongjumped from interrupt CTRL Z 20

Demarrage du sleep

Fin du sleep

bytes reçus:268 Type:2 Numero:1

Received packet from 127.0.0.1:1400

Ouverture reussie VoituresHV

Trouve Ford,GT 50

Record Ecrits 1

Ouverture reussie de FactureHV

Mise à jour du Nombre réussie

bytes écrits:268

bytes reçus:268 Type:2 Numero:1

Received packet from 127.0.0.1:1400

Ouverture reussie VoituresHV

Trouve Ford,GT 50

Record Ecrits 1

Ouverture reussie de FactureHV

Mise à jour du Nombre réussie

bytes écrits:268

bytes reçus:268 Type:2 Numero:1

Received packet from 127.0.0.1:1400

Ouverture reussie VoituresHV

Trouve Ford,GT 50

Record Ecrits 1

Ouverture reussie de FactureHV

Mise à jour du Nombre réussie

bytes écrits:268

() Champ Perso**

Consulter les factures créées avec admin

```
vanstap@ubuntu:~/lib20200630/Step5AB$ ./admin
-----
1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----
2
Ouverture reussie
Ref Marque          Modele          Nombre
1 Ford              Fiesta          0
2 Ford              Focus           1
3 Ford              Mondeo         2
4 Ford              Mustang        2
5 Ford              GT 49          3
6 Ford              GT 50          3
-----
1) Ajout
2) Voitures
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
-----
5
Ouverture reussie
1 toto              1 1 0
2 toto              1 1 0
3 toto              1 1 0
4 Super Super      1 6 0
5 Super Super      1 6 0
6 Super Super      1 6 0
7 Super Super      1 6 0
8 spiderman du lundi      1 6 0
9 spiderman du lundi      1 6 0
10 spiderman du lundi     1 6 0
```

Le serveur à cette étape ne gère pas les doublons, ce sera corrigé plus tard

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 ses transactions soient également numérotées.

Intégrer le timeout n'est pas nécessaire mais il faut intégrer au minimum le code de la gestion des doublons

Dans l'exemple suivant, on fait en premier un achat et on force les timeout en faisant ctrl z sur le serveur.

Le client affiche le résultat. Je fais ensuite une recherche sur la référence un pour connaître le nombre d'articles restants. Le client reçoit la réponse et traite les doublons.

```
vanstap@ubuntu:~/lib20200630/Step5AB$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
-----
1) Demander une reference
2) Acheter une Voiture
3) Quitter
Choix :2
NomClient :Madona
Reference :7
Nombre:1
Envoi de 268 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 268 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 268 bytes
bytes:268
Achat Reussi Facture : 11
-----
1) Demander une reference
2) Acheter une Voiture
3) Quitter
-----
Choix :1
Reference :7
Envoi de 268 bytes
doublon 0 !!!!!
doublon 0 !!!!!

bytes:268
Pour 7 Ford,GT 51 Nombre: 4 (**)
```


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 un véhicule commandé, on va retirer quatre voiture du stock 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. 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 stock du véhicule. Si on trouve une correspondance, le serveur retourne au client le numéro de facture déjà généré et ne touche pas au stock du véhicule

La date sera stockée dans un entier sous forme simplifiée. Le 18 janvier à 11 heures donnera 011811

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

L'affichage du client

```
vanstap@ubuntu:~/lib20200630/Step6$ ./cli 127.0.0.1 1400 127.0.0.1 1300
```

```
port 1400
```

```
CreateSockets 3
```

- ```

1) Demander une reference
2) Acheter une Voiture
3) Quitter

```

```
Choix :1
```

```
Reference :6
```

```
Envoi de 268 bytes
```

```
bytes:268
```

```
Pour 6 Ford,Puma Nombre: 19 (**)
```

- ```
-----  
1) Demander une reference  
2) Acheter une Voiture  
3) Quitter  
-----
```

```
Choix :2
```

```
NomClient :spider solitaire
```

```
Reference :6
```

```
Nombre:1
```

```
Date:11112
```

```
Envoi de 268 bytes
```

```
error sur receive:: Interrupted system call
```

```
rc -1 errno:4
```

```
Envoi de 268 bytes
```

```
error sur receive:: Interrupted system call
```

```
rc -1 errno:4
```

```
Envoi de 268 bytes
```

```
bytes:268
```

```
Achat Reussi Facture : 2
```

- ```

1) Demander une reference
2) Acheter une Voiture
3) Quitter

```

```
Choix :
```

### L'affichage du serveur

```
vanstap@ubuntu:~/lib20200630/Step6$./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
CreateSockets 3
bytes:268 Type:1 Numero:0
A rechercher 6
Ouverture reussie VoituresHV
res :1 Ford,Puma Nombre 19 (**)
bytes:268
^Zlongjumped from interrupt CTRL Z 20
Demarrage du sleep
Fin du sleep
bytes:268 Type:2 Numero:1
Ouverture reussie FactureHV
On va tenter de réserver
Ouverture reussie VoituresHV
Trouve Ford,Puma
Record Ecrits 1
Reservation Reussie
Ouverture reussie de FactureHV
 Mise à jour du stock réussie
bytes:268
bytes:268 Type:2 Numero:1
Ouverture reussie FactureHV
Doublon
bytes:268
bytes:268 Type:2 Numero:1
Ouverture reussie FactureHV
Doublon
bytes:268
```

**(\*\*) champ personnel**

## Le programme admin \*\*

```
vanstap@ubuntu:~/lib20200630/Step6$./admin
```

- ```
-----  
1) Ajout  
2) Voitures  
3) Recherche  
4) Achat  
5) Factures  
6) A propos  
7) exit  
-----
```

```
2
```

```
Ouverture reussie
```

Ref	Marque	Modele	Nombre
1	Ford	Fiesta	0
2	Ford	Focus	1
3	Ford	Mondeo	2
4	Ford	Mustang	2
5	Ford	GT 49	3
6	Ford	Puma	18

7) Communication entre un client Linux et un serveur sun

Documentation

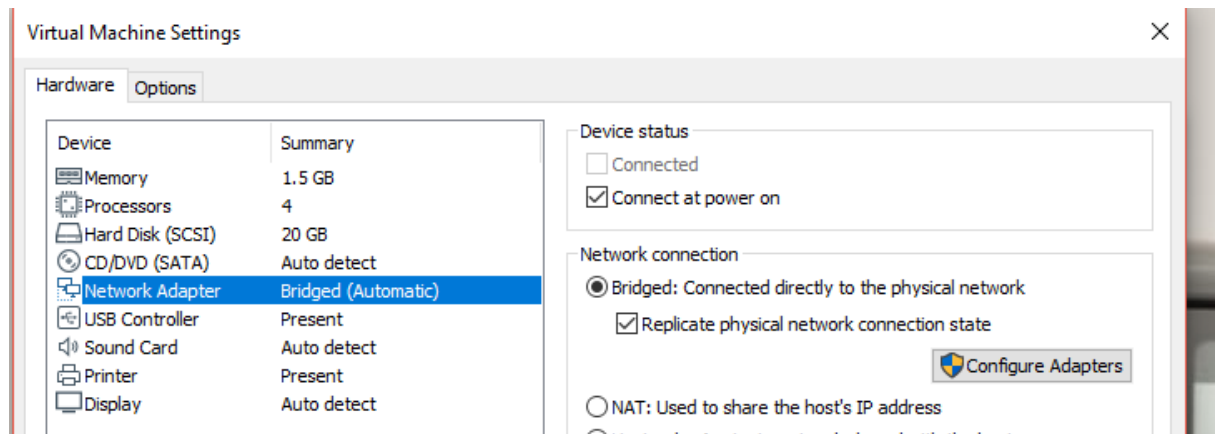
Voir Tome4P, exemple ex08.

Analyse

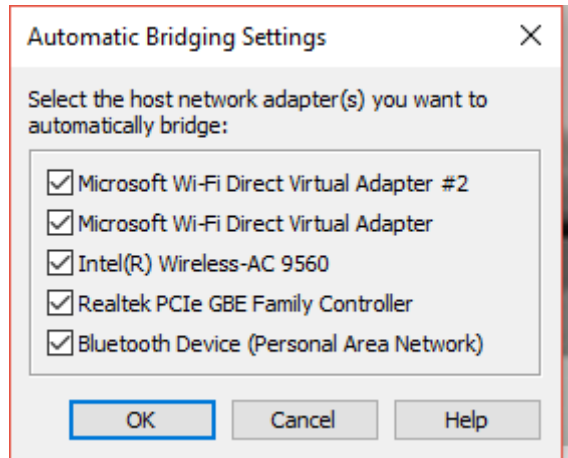
On vous demande de démarrer un serveur sur machine sun et de faire la recherche à partir d'un client linux. Les entiers doivent être convertis avec les fonctions hton et ntoh.

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 wireshark peuvent engendrer des problèmes

Obtenir les ips des machines

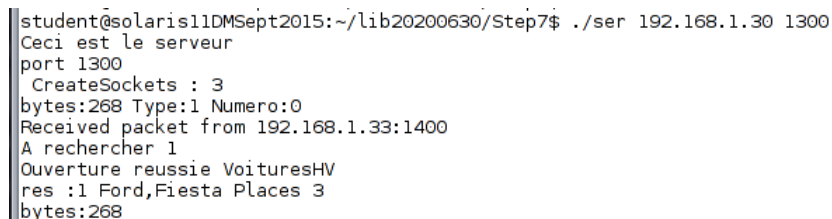
```
student@solaris11DMSept2015:~$ ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index
1
    inet 127.0.0.1 netmask ffffffff
net0: flags=100001004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,PHYSRUNNING> m
tu 1500 index 2
    inet 192.168.1.30 netmask fffffff0 broadcast 192.168.1.255
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index
1
```

```
vanstap@ubuntu:~/lib20200630/Step6$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.33 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2a02:a03f:4353:c900:643d:a5eb:27c9:7b01 prefixlen 64 scopeid 0x0
```

Faire un ping pour vérifier que la connexion fonctionne bien

```
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=255 time=0.552 ms
64 bytes from 192.168.1.30: icmp_seq=2 ttl=255 time=0.640 ms
64 bytes from 192.168.1.30: icmp_seq=3 ttl=255 time=0.423 ms
64 bytes from 192.168.1.30: icmp_seq=4 ttl=255 time=0.550 ms
64 bytes from 192.168.1.30: icmp_seq=5 ttl=255 time=0.403 ms
```

Le serveur sous sun



```
student@solaris11DMSept2015:~/lib20200630/Step7$ ./ser 192.168.1.30 1300
Ceci est le serveur
port 1300
CreateSockets : 3
bytes:268 Type:1 Numero:0
Received packet from 192.168.1.33:1400
A rechercher 1
Ouverture reussie VoituresHV
res :1 Ford,Fiesta Places 3
bytes:268
```

Le client sous linux

```
vanstap@ubuntu:~/lib20200630/Step7$ ./cli 192.168.1.33 1400 192.168.1.30 1300
port 1400
CreateSockets 3
-----
1) Demander une reference
3) Quitter
-----
Choix :1
Reference :1
Envoi de 268 bytes

bytes:268
Pour 1 Ford,Fiesta Nombre : 3
```

Cela ne fonctionne pas

Vérifiez bien les ips et les ports

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 multclients 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)

En partant du point 3, 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) ;
```