Enoncé de Réseau & Programmation Réseau V 1.4
(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
lib2019Etud.tar	Sun ou Linux

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

3) Lib2019Etud\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 Cinéma

La librairie libSerS**HV** 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 les films qui sont stockés dans un fichier **Seances*HV***. Il permettra aussi de facturer la vente de tickets 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 Séance dans le fichier ***SeancesHV*** créé par admin ou une vente qui est stockée alors dans le fichier ***FactureHV***
cli	Ce programme permet à l'utilisateur de consulter les séances disponibles et d'acheter des tickets. 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	19/11 A présenter avant l'examen
2	Un Client et un seul serveur la requête recherche	26/11 A présenter avant l'examen
3	Un Serveur et plusieurs clients	10/12 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. **Pénalité si recopie ou non remise du dossier**	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	A présenter avant l'examen
2	Un Client et un seul serveur la requête recherche	A présenter avant l'examen
3	Un Serveur et plusieurs clients	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. **Pénalité si recopie ou non remise du dossier**	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	Studio		Sony, Disney
2	Genre		Policier, guerre , Science-Fiction
3	Duree	Durée en minutes	100 , 200
4	Acteur principal		Tom Cruise, daniel Craig
5	Actrice principale		Scarlett Johanson
6	Technologie		Noir & blanc, Couleur ; 3D
7	Réalisateur		Rupert Sanders , George Lucas
8	Lieu de tournage		USA, France
9	Année	Année de tournage	1989
10			

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 **lib2019Etud\step0**
Copier le contenu de ce dossier dans **lib2019Etud\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

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

```
#ifdef DATAH

#else


#define DATAH

struct SeanceHV {
       int  Reference ;
            char Film[40] ;
            int  Places ;
       char Date[10] ;
           } ;

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 séances 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 lib2016.c

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

int Recherche retourne 1 si la recherche a réussi.
Char *Nom`Fichier` désigne chaque fois le nom du fichier qui contient les Seances,
Reference : le numéro de la séance à chercher
struct ArticleHV* 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

```
vanstap@ubuntu:~/lib2019/Step1$ ./admin
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
2
Ouverture reussie
Ref Film                                Places
Record lu 1 et Position actuelle dans le fichier 60
1   Terminator                           30    (**)
Record lu 1 et Position actuelle dans le fichier 120
2   Star wars 1                          20    (**)
Record lu 1 et Position actuelle dans le fichier 180
3   Star wars 2                          20    (**)
Record lu 1 et Position actuelle dans le fichier 240
4   Star Wars 3                          25
```

```
Record lu 1 et Position actuelle dans le fichier 300
5   Star Wars 4                    24    (**)
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
2
Ouverture reussie
Ref Film                           Places
Record lu 1 et Position actuelle dans le fichier 60
1   Terminator                     30                (**)
Record lu 1 et Position actuelle dans le fichier 120
2   Star wars 1                    20                (**)
Record lu 1 et Position actuelle dans le fichier 180
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
3
Saisie Reference:1
Ouverture reussie SeancesHV
Ref Film                           Places
1   Terminator                     30                (**)
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
```

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 :

```
# cphex\makefile

LIBS=
all:        admin    cli         ser         udplib.o LibSerHV.o


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

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

LibSer est à completer avec vos initiales

Si vous travaillez avec SUN ; completez 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 {
    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 ;
 int Date ;
 int Reference ;
 int Places ;
 int Prix ;
 char Film[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:~/lib2019/Step2$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
Saisie numero Seance:1
Envoi de 188 bytes

 bytes:188
Film:Terminator (**)
```

L'affichage du serveur

```
vanstap@ubuntu:~/lib2019/Step2$ ./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
 CreateSockets : 3
bytes:188 Reference:1
Ouverture reussie SeancesHV
res :1 Reference:Terminator  (**)
bytes:188
```

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 le numéro de séance à chercher et envoie l'identifiant au serveur par réseau qui répond en fournissant le nom du film , le nombre de places restantes et le champ perso.

```
vanstap@ubuntu:~/lib2019/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 188 bytes

 bytes:188
Pour 1 Film:Terminator Places: 30   (**)
```

```
vanstap@ubuntu:~/lib2019/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 :1
Envoi de 188 bytes

 bytes:188
Pour 1 Film:Terminator Places: 30 (**)
```

(**) 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 Question:
        fprintf(stderr,"A rechercher %d \n", UneRequeteR.Reference ) ;
              res = RechercheHV("SeancesHV",UneRequeteR.Reference ,&UnRecord) ;
              fprintf(stderr,"res :%d  Film:%s Places %d\n",res,UnRecord.Film,UnRecord.Places ) ;
              /* reponse avec psor qui contient toujours l'adresse du dernier client */
        strncpy(UneRequeteE.Film,UnRecord.Film,sizeof(UneRequeteE.Film)) ;
        UneRequeteE.Places = UnRecord.Places ;
        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:~/lib2019/Step3$ ./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
 CreateSockets : 3
bytes:188 Type:1 Numero:0
Received packet from 127.0.0.1:1400
A rechercher 1
Ouverture reussie SeancesHV
res :1  Film:Terminator Places 30 (**)
bytes:188
bytes:188 Type:1 Numero:0
Received packet from 127.0.0.1:1500
A rechercher 1
Ouverture reussie SeancesHV
res :1  Film:Terminator Places 30 (**)
bytes:188
```

(**) 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**. J'achète des articles pour un numéro dz). **Le programme modifie le stock de l'article en retirant le nombre de places commandées** et ajoute une facture au nom de l'acheteur dans le fichier facture

```
vanstap@ubuntu:~/lib2019/Step4A$ ./admin
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
4
NomClient :Zorro le héros
Reference :1
Places:1
Ouverture reussie SeancesHV
 Trouve Reference Terminator  (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour du Fichier FactureHV réussie
```

(**) est le champ personnel.

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

```
----------------------
1) Ajout
2) Seances
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   Toto                                1   2   0
Record lu 1 et Position actuelle dans le fichier 112
2   Zorro le héros                          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 ReservationHV(char* NomFichier,int Reference ,int Places ) ;
```

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

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

```
int FacturationHV(char NomFichier[80], char NomClient[60], int Date,int Places,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 &

```
vanstap@ubuntu:~/lib2019/Step4B$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :2
NomClient :Superman ce heros
Reference :1
Places:1
Envoi de 188 bytes

 bytes:188
Achat Reussi Facture : 4
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :
```

Le fonctionnement illustré du serveur

```
vanstap@ubuntu:~/lib2019/Step4B$ ./ser 127.0.0.1 1300
Ceci est le serveur
port 1300
 CreateSockets : 3
bytes:188 Type:2 Numero:0
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
 Trouve Reference Terminator  (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour du nombre de places réussi
bytes:188
```

(**) Ne pas oublier champ personnel

Vérification du résultat avec admin

```
 vanstap@ubuntu:~/lib2019/Step4B$ ./admin
----------------------
1) Ajout
2) Seances
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   Toto                                    1   2   0
Record lu 1 et Position actuelle dans le fichier 112
2   zorro                                   2   1   0
Record lu 1 et Position actuelle dans le fichier 168
3   spiderman                                   1   1   0
Record lu 1 et Position actuelle dans le fichier 224
4   Superman ce heros                               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:~/lib2019/Step5AB$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :2
NomClient :superzero
Reference :1
Places:1
Envoi de 188 bytes
error sur receive:: Interrupted system call   ← 1 retransmission
rc -1 errno:4
Envoi de 188 bytes
error sur receive:: Interrupted system call <- 2 retransmission
rc -1 errno:4
Envoi de 188 bytes
error sur receive:: Interrupted system call   <- 3 retransmission
rc -1 errno:4
Envoi de 188 bytes

 bytes:188
Achat Reussi Facture : 8  <- Le serveur a répondu
```

```
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :2
NomClient :superHeros II     <- nouvel achal
Reference :2
Places:1
Envoi de 188 bytes
doublon 0 !!!!!              <- doublon du précédent achat auquel le serveur à répondu
doublon 0 !!!!!              <- doublon du précédent achat auquel le serveur à répondu
doublon 0 !!!!!              <- doublon du précédent achat auquel le serveur à répondu

 bytes:188
Achat Reussi Facture : 12
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :
```

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:~/lib2019/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:188 Type:2 Numero:0
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
 Trouve Reference Terminator  (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour des places réussie
bytes:188
bytes:188 Type:2 Numero:0
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
 Trouve Reference Terminator (**)
Record Ecrits 1
Ouverture reussie de FactureHV
```

```
Mise à jour des places réussie
bytes:188
bytes:188 Type:2 Numero:0
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
 Trouve Reference Terminator (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour des places réussie
bytes:188
bytes:188 Type:2 Numero:0
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
 Trouve Reference Terminator (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour des places réussie
bytes:188
bytes:188 Type:2 Numero:1
Received packet from 127.0.0.1:1400
Ouverture reussie SeancesHV
Record lu 1 et Position actuelle dans le fichier 60
 Trouve Reference Star wars 1 (**)
Record Ecrits 1
Ouverture reussie de FactureHV
Mise à jour des places réussie
bytes:188
```

(**) Champ Perso

Consulter les factures crées avec admin

```
vanstap@ubuntu:~/lib2019/Step5AB$ ./admin
----------------------
1) Ajout
2) Seances
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   Toto                                1   2   0
Record lu 1 et Position actuelle dans le fichier 112
2   zorro                               2   1   0
Record lu 1 et Position actuelle dans le fichier 168
3   spiderman                             1   1   0
```

```
Record lu 1 et Position actuelle dans le fichier 224
4  zozo                                  1  1  0
Record lu 1 et Position actuelle dans le fichier 280
5  eeeeee                                1  1  0
Record lu 1 et Position actuelle dans le fichier 336
6  eeeeee                                1  1  0
Record lu 1 et Position actuelle dans le fichier 392
7  eeeeee                                1  1  0
Record lu 1 et Position actuelle dans le fichier 448
8  superzero                             1  1  0
Record lu 1 et Position actuelle dans le fichier 504
9  superzero                             1  1  0
Record lu 1 et Position actuelle dans le fichier 560
10 superzero                             1  1  0
Record lu 1 et Position actuelle dans le fichier 616
11 superzero                             1  1  0
Record lu 1 et Position actuelle dans le fichier 672
12 superHeros II                         1  2  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 connaitre le nombre d'articles restants. Le client reçoit la réponse et traite les doublons.

```
vanstap@ubuntu:~/lib2019/Step5AB$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :1
Reference :1
Envoi de 188 bytes

 bytes:188
Pour 1 Film:Terminator Places: 19
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :2
NomClient :JOJO 2
Reference :1
Places:1
Envoi de 188 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 188 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 188 bytes

 bytes:188
Achat Reussi Facture : 13
------------------------------
```

```
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :1
Reference :2
Envoi de 188 bytes
doublon 1 !!!!!
doublon 1 !!!!!

 bytes:188
Pour 2 Film:Star wars 1 Places: 18 (**)
------------------------------
1) Demander une reference
2) Acheter un ticket
3) Quitter
------------------------------
Choix :
```

(**) 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 une place , on va retirer quatre fois la place alors que le client n'en a demandé qu'une place.

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 nombre de places de la séance. Si on trouve une correspondance, le serveur retourne au client le numéro de facture déjà généré et ne touche pas au nombre de places de la séance

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:~/lib2019/Step6$ ./cli 127.0.0.1 1400 127.0.0.1 1300
port 1400
CreateSockets 3
------------------------------
1) Demander une reference
2) Acheter une Place
3) Quitter
------------------------------
Choix :1
Reference :1
Envoi de 188 bytes

 bytes:188
Pour 1 Film:Terminator Places: 22
------------------------------
1) Demander une reference
2) Acheter une Place
3) Quitter
------------------------------
Choix :2
NomClient : Spiderman 69
Reference :1
```

```
Places:1
Date:110203
Envoi de 188 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 188 bytes
error sur receive:: Interrupted system call
rc -1 errno:4
Envoi de 188 bytes

 bytes:188
Achat Reussi Facture : 9
------------------------------
1) Demander une reference
2) Acheter une Place
3) Quitter
------------------------------
Choix :2
NomClient :zozo 12
Reference :1
Places:1
Date:120203
Envoi de 188 bytes
doublon 1 !!!!!
doublon 1 !!!!!

 bytes:188
Achat Reussi Facture : 10
------------------------------
1) Demander une reference
2) Acheter une Place
3) Quitter
------------------------------
Choix :
```

L'affichage du serveur

```
vanstap@ubuntu:~/lib2019/Step6$ ./ser  127.0.0.1 1300
Ceci est le serveur
port 1300
CreateSockets 3
bytes:188 Type:1 Numero:0
A rechercher 1
Ouverture reussie SeancesHV
res :1 Reference:Terminator Places 22  (**)
bytes:188
^Zlongjumped from interrupt CTRL Z  20
Demarrage du sleep
Fin du sleep
bytes:188 Type:2 Numero:1
```

```
Ouverture reussie FactureHV
On va tenter de réserver
Ouverture reussie SeancesHV
 Trouve Reference Terminator (**)
Record Ecrits 1
Reservation Reussie
Ouverture reussie de FactureHV
            Mise à jour du stock réussie
bytes:188
bytes:188 Type:2 Numero:1
Ouverture reussie FactureHV
Doublon
bytes:188
bytes:188 Type:2 Numero:1
Ouverture reussie FactureHV
Doublon
bytes:188
bytes:188 Type:2 Numero:2
Ouverture reussie FactureHV
On va tenter de réserver
Ouverture reussie SeancesHV
 Trouve Reference Terminator (**)
Record Ecrits 1
Reservation Reussie
Ouverture reussie de FactureHV
            Mise à jour du stock réussie
bytes:188
```

(**) champ personnel

Le programme admin **

```
 vanstap@ubuntu:~/lib2019/Step6$ ./admin
----------------------
1) Ajout
2) Seances
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   Toto                                1   2   0
……
9   Spiderman 69                        1   1   110203
Record lu 1 et Position actuelle dans le fichier 560
```

```
10  zozo 12                          1   1   120203
----------------------
1) Ajout
2) Seances
3) Recherche
4) Achat
5) Factures
6) A propos
7) exit
----------------------
```

7) Le calcul du crc

Documentation

Voir Tome 21

Analyse

Ajouter à la structure requête du client , un champ crc. Le crc doit être calculé avant envoi, on affichera sa valeur avant envoi **qui doit être zéro** et sa valeur à la réception. La politique pour une trame défectueuse est de la jeter sans acquittement. Consulter l'exemple suivant

```
#include "stdio.h"
#include <strings.h>

unsigned short cksum(void *ipt, int len)
{
long sum = 0 ; /* assume 32 bit long, 16 bit short */
unsigned short *ip ;
ip = ipt ;
while ( len > 1 ) {
    sum += *(ip)++ ;
    if ( sum & 0x80000000 ) /* if hogh-order bit set, fold */
    sum = (sum & 0xFFFF ) + (sum >> 16 ) ;
    len -= 2 ;
}
if (len) /* take care of left over byte */
   sum += (unsigned short) *(unsigned char *) ip ;
while( sum>>16 )
  sum = (sum & 0xFFFF) + ( sum >> 16 ) ;
return ~sum ;
}

struct Requete{
     int val1 ;
     int val2 ;
     int val3 ;
     int val4 ;
     unsigned short s0 ; /* Bidon sinon problème de checksum */
     unsigned short crc ;
};

void main()
{
 struct Requete E;
 bzero(&E, sizeof( struct Requete)) ;
 printf("La taille de la structure %ld %ld\n",sizeof( struct Requete) ,sizeof(short)) ;
 E.val1 = 1 ; E.val2 = 2 ; E.val3 = 3 ; E.val4 = 4 ; E.s0 = 0 ; E.crc = 0;
 E.crc = cksum(&E,sizeof(struct Requete)) ;
 printf("Vérification du checksum %d \n",cksum(&E,sizeof(struct Requete))) ;
}
```

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 multiclients 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) ;
```