# Enoncé de Laboratoire Réseau & Programmation 2 Gestion &Telecom & indus 2010-2011

## Objectif du laboratoire : Parking

Ecrire des programmes en langage C qui vont gérer la réservation de places pour un parking

Vous aurez à créer les programmes suivants :

Un utilitaire

***CreaTransacServeur***: Crée le fichier de transaction du serveur

Les clients :

***BarrièreEntrée*** : Permet de faire entrer des voitures dans le parking. Délivre le ticket

***BornePayement*** : Permet de payer les tickets avant de reprendre son véhicule.

***BarrièreSortie :*** Permet de sortir du parking

Le serveur :

***Serveur*** : Gère le fichier des transactions et coordonner l’accès à ce fichier par les clients.

***C’est une évaluation continue avec des échéances. Voir fin de ce document***

## La plateforme de développement

Les exemples ont été développés sur Ubuntu (disponible en live cd ou installation native). Vous pouvez réaliser vos programmes sur une autre plateforme (Sun, un document d adaptation est fourni) à condition

1. Qu'elle soit capable de faire fonctionner tout les exemples.
2. Que la plateforme soit accessible depuis l’école.
3. Que votre code reste portable ce qui permet de continuer votre travail sur une autre plateforme en cas de panne de votre plateforme.

***Quelque soit la plateforme, vous devrez utiliser les ports donnés dans le fichier Excel qui sera fournit sur l’école virtuelle.***

**Vous êtes tenu de gardez de copies de sauvegardes**

## Les outils à utiliser

Tous les outils sont disponibles sur ma page dans le centre de ressources de EV.

Vous devez disposer de trois fichiers pour commencer votre travail.

1. Dans répertoire Réseau Programmation\notes, télécharger le ***tome R4Prog.***
2. Dans les répertoires Librairies, télécharger les librairies LIB201032gg (testées sous ***ubuntu***). Si vous choisissez de travailler sur ***sun***, télécharger le fichier ***directives*** qui vous permettra de faire les adaptations correspondantes.
3. Dans le répertoire Réseau Programmation\exercices, vous trouverez cet énoncé.

## Considérations préalables

Sur le canal d’erreur on écrira un fichier de log qui devra contenu au minimum le nombre de bytes lus et écrits ainsi que le contenu de la structure requête pour chaque requête lue ou écrite. ***Lire la page 7 du tome 4***

***Chaque programme devra comporter un menu avec vos noms & prénoms.***

## Etape 0 : Ecrire une gestion de fichiers

### Les sources à consulter

***Utiliser les fonctions fichiers de haut niveau***

Répertoire ***Fichiers.*** Dans les notes, les sources sont en ***annexe***.

Télécharger aussi l’exemple ***AccesDirect.c***

### Objectif

Cette étape vous permet de vous concentrer sur les erreurs de logique afin de ne plus que vous concentrer sur les erreurs réseaux par la suie.

Créer un programme ***CreaTransacServeur*** qui va permettre de gérer le fichier transaction dont l’objectif est de contenir les tickets délivrés par les barrières d’entrées.

Le menu de cette application comprend

1. Créer le fichier transactions
2. Réserver un ticket
3. Payer le ticket
4. Sortir du Parking
5. Lister le contenu du fichier
6. Quitter

Vous aurez besoin d’un fichier transactions

Comportant les informations suivantes (tous les champs sont des long int)

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Fonction |
| IP : | int | IP de l’émetteur |
| Port | int | Port de l émetteur |
| NumTransa***c*** | int | Transaction du client |
| Heure | int | Heure de l’action (Chiffre entre 0 et 24) |
| Places Libres | int | Le nombre de places restantes |
| Numticket : | int | Le dernier Numéro de ticket attribué |
| ***Action*** | int | 1 ENTETE, 2 RESERVATION, 3 PAYEMENT, 4 SORTIE |

Les champs avec \* nous ne les utiliseront pas dans ce premier exercice mais prévoyez les déjà pour la suite

Pour raisons de clarté, nous ne considérons que les champs suivants en premier

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |

1. Créer le fichier transactions

On ajoute le record suivant dans le fichier (tous les champs sont des long int)

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 99 | 0 | 0 | ENTETE |

NB : si le fichier existe déjà, on ne fait rien, on teste seulement la présence de ENTETE pour vérifier sa validité

1. Réserver un ticket en entrant une heure.

Lire le nombre de Places Libres dans le Premier record du fichier.

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 99 | 0 | 0 | ENTETE |

Remplacer ce record par le suivant :

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 98 | 1 | 0 | ENTETE |

Ceci crée un nouvel enregistrement à la fin du fichier, on suppose que l’on rentre à 10 H

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 98 | 1 | 10 | RESERVATION |

1. Payer le ticket

Le programme demande le numéro de ticket à payer & L’heure de payement

Recherche le record dans le fichier en commençant de préférence par la fin (moins d’accès) si un record existe avec Action = RESERVATION

Si c’est le ticket 1 AJOUTER le record suivant à la fin de fichier. 11 Heure à laquelle on paye.

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 98 | 1 | 11 | PAYEMENT |

1. Sortir du Parking

Le programme demande le numéro de ticket et l’heure de sortie. (identique à l’heure de payement car les parkings n’accordent souvent que 15 minutes pour sortir)

Recherche le record PAYEMENT dans le fichier en commençant de préférence par la fin (moins d’accès). Ne prendre en compte dans la recherche que les records avec le flag 0.

Si c’est le ticket 1

AJOUTER Le RECORD suivant en fin de fichier

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 98 | 1 | 11 | SORTIE |

Modifier le Record avec ENTETE. Une place devient libre maintenant

|  |  |  |  |
| --- | --- | --- | --- |
| PlacesLibres | NumTicket | Heure | Action |
| 99 | 1 | 0 | ENTETE |

5 )Lister le contenu du fichier

## Etape 1 : Echanger une structure entre un client et un serveur

### Les sources à consulter.

cphexS , ***exemple 2***

### Objectif

Vous devez créer les programmes suivants

***BarrièreEntrée*** : Permet de faire entrer des voitures dans le parking. Délivre le ticket

Au niveau de la BarrièreEntree, pour simuler la pression du bouton de demande de tickets, entrez l’heure (un entier) puis enter.

La requête est envoyée à ***Serveur***  via une structure C qui sera commune aux deux programmes (utiliser un .h commun). La structure peut avoir le format suivant

Struct Requete {

Int Code ;

Int Heure ;

Int NumTransaction ; /\* sera utilisé plus tard \*/

Long int IP ; /\* sera utilisé plus tard \*/

Int port ; /\* sera utilisé plus tard \*/

Int NumTicket; /\* Le numero de ticket \*/

}

Chaque demande de ticket entraine l’ajout dans le fichier transaction du serveur selon les règles indiquées à l’étape0.

La réponse est envoyée au client qui affichera le numéro de ticket de manière bien visible. La saisie d’une heure permet de demander un autre ticket

## Etape 2 : Echanger une structure entre plusieurs client et un serveur : Le programme de réservation (1 version)

### Les sources à consulter

cphexser2 ou exemple 3 dans les notes

remarque dans l’exemple, on utilise toujours des chaînes de caractères. Pour ce point, vous devrez toujours utiliser des structures comme pour l’étape 1.

***Attention*** au niveau du serveur utiliser maintenant la primitive ***repondre*** au lieu de verscouchesphysique

On vous demande d’utiliser ***AttendreEvenement*** au niveau du serveur.

### Objectif

Plusieurs barrières d’entrées peuvent être connectées au serveur.

Le serveur doit toujours afficher la liste des 20 dernières requêtes et ne pas bloquer sur une requête clavier (***attendreevenement)***

***Bonus***

Pour ceux qui désirent optimiser, vous pouvez placer un fork au niveau du serveur quand il répond à une requête d'affichage. Le fork doit se faire après la réception de la requête du client. Après le fork, le sous processus fournira la liste des films, le cas de l’option 1, ou des séances dans le cas de l’option 2.

Ceci évite de monopoliser trop longtemps un serveur pour un seul client.

***Remarque :*** si vous avez des problèmes avec des caractères qui restent dans le tampon, utilisez la commande tcflush.

## Etape 3 : La gestion du clavier sur le serveur & client Le programme de réservation (2 version)

### Les sources à consulter

cphexser3 ou etape 5

### Objectif

Chaque client a son propre fichier de transactions

On vous demande maintenant d intégrer ***attendreEvenement*** au niveau du client. Regardez attentivement l’exemple du client. Le client ne peut être bloqué ni sur une requête client ni sur une requête réseau. ***Le client ne peut faire un fgets tant qu’attendreévenement n’a pas retourné un évènement clavier***

Le Client gère maintenant un fichier transactions dans lequel il stocke un numéro de transaction et le numéro de ticket reçu.

Attention le fichier transaction est propre à chaque client et ne peut être partagé entre les clients (Il est conseillé de générer le nom du fichier selon le port client). Le numéro de transaction est propre à chaque client (Si un client fait une transaction, cela ne change pas le numéro de transaction de l autre client)

Par contre la numérotation des tickets est gérée par le serveur et il ne peut y avoir deux clients qui reçoivent le même numéro de ticket.

Il ne peut y avoir de trou dans la numérotation des tickets et de transaction.

SI je fais CTRL C sur le serveur ou sur un des clients, quand aucune transaction n’est encours, il ne peut y avoir de perte de ticket ou de numéro de transaction.

***A tout moment*** sur le programme réservation, je peux consulter les 20 dernières requêtes reçues. L’opération ne doit pas bloquer le client (uniquement le temps d’affichage des 20 lignes)

***A tout moment*** sur le programme serveur je peux consulter les 20 dernières requêtes de réservation reçues. L’opération ne doit pas bloquer le serveur (uniquement le temps d’affichage des 20 lignes)

***Bonus :*** Si le client n’encode rien après 60 secondes, revenir au menu principal.

## Etape 4 : la gestion des timers et doublons

### Les sources à consulter

cphexser4 ou étape 6

### Objectif

Quand le programme de réservation de Ticket ne reçoit pas une réponse après 10 secondes, il doit retransmettre la requête. Tant qu’il ne reçoit pas de réponse, la requête est transmise quatre fois (le nombre de retransmission doit être affiché à l’écran soit un minimum ou trois maximum)

Au niveau du serveur, vous devez rajouter une option sleep au menu. Quand on l’active, on indique un temps en seconde le serveur appelle la fonction système unix sleep et qui endort le processus le temps indiqué.

***Les tests***

Pour tester les timers, on fera un premier test avec ***l’option sleep***. Le sleep doit être choisit de manière à ce que les timers des clients se déclenchent. Après la sortie du sleep du serveur, les tickets doivent être délivrés correctement aux clients (sans trou dans la numérotation)

## Etape 4B : la gestion des timers et doublons

### Les sources à consulter

cphexser4 ou étape 6

### Objectif

Dans cette étape en plus des timers de l’étape précédente, il faut maintenant gérer les doublons. Suite aux retransmissions, plusieurs tickets peuvent être délivrés pour une seule demande ce qui n’est pas correct.

Pour identifier de manière unique chaque transaction du programme réservation, on utilise **l IP et le port du client** (Fonctions IPDistante, PortDistant) **et un numéro de transaction** qui est propre à chaque client et incrémenté à chaque envoi de données. Ces informations sont transmises par le client au serveur

Si le serveur reçoit une transaction, il doit vérifier si elle a déjà été traitée en consultant son fichier et en comparant sur les champs **en gras souligné**. Si ce n’est pas le cas, un nouveau numéro de ticket doit être généré et ajouté au fichier transaction du serveur. Si cette transaction a déjà été traitée, le serveur retourne au client le numéro de ticket qu’il avait déjà fournit et n’ajoute pas de nouvelle transaction à son fichier transaction.

Si le client reçoit deux fois une réponse à une requête, il doit jeter le doublon. Pour cela Il lui suffit de consulter simplement son fichier transaction. Si la transaction n’est pas présente, elle est ajoutée et à l’écran on afficher le numéro de ticket. Si elle est déjà présente on jette le doublon

***Les tests***

Le test du sleep est toujours valable.

On fera un second test avec ***l’option ctrl c*** sur le serveur. Avant que le serveur redémarre vous attendez que les timers des clients se déclenchent. Après redémarrage du serveur, les tickets doivent être délivrés correctement aux clients (sans trou dans la numérotation)

## Etape 5 : la borne de payement

### Les sources à consulter

cphexser4 ou étape 6

### Objectif

1. Demander un ticket d entrée à la barrière d entrée puis garer son véhicule
2. Avant de reprendre son véhicule, payer le ticket à la borne de paiement
3. Prendre son véhicule et sortir du parking par la barrière de sortie

Le menu de la borne de payement propose

1. Encoder le ticket et heure d'entrée
2. Payer le ticket
3. Quitter
4. Le client encode son numéro de ticket puis l’heure de sortie  
   L’option 2 n’est pas disponible. La borne vérifie en premier que ce ticket existe et propose la somme.
5. . L option 1 n’est plus disponible. Le client paye et le serveur Ajoute le record correspondant au Fichier (voir Etape 0).

Un fichier transaction doit être géré au niveau de la borne et les timers doivent fonctionner correctement en cas de sleep du serveur ou ctrl C

***Bonus*** : Un timer peut être ajouté pour éviter qu'un client ne monopolise une borne.

## Etape 6 : la barrière de sortie

### Les sources à consulter

cphexser4 ou étape 6

### Objectif

Le Client vient devant la barrière de sortie. Il encode son numéro de ticket et l’heure de sortie. La barrière envoie via le réseau au serveur la requête. Le serveur vérifie si le ticket est bon et ajoute le record de sortie à son fichier de transaction. Une place est libérée.

Un fichier transaction doit être géré au niveau de la borne et les timers doivent fonctionner correctement en cas de sleep du serveur ou ctrl C

## Etape 7 : l algorithme de calcul crc

### Les sources à consulter

Voir Tome 2

Objectif

Ajouter à la structure requête, un champ crc. Le crc doit être calculé avant envoi puis vérifié à la réception. La politique pour une trame défectueuse est de la jeter sans acquittement.

## Etape 8 : le serveur multi port

### Les sources à consulter

cphexser2b

### Objectif

Le serveur écoute chaque requête sur un seul port.

Dans la version multi port, le serveur écoute les barrières d’entrées sur un port, les bornes de payement sur un autre port et la barrière de sortie sur un troisième. Donc au lieu de démarrer le serveur sur un seul port, on spécifiera deux ports supplémentaires.

## Organisation du travail

C’est un dossier pour deux ou seul (voir avec le prof). Vous êtes tenus de respecter les échéances ci-dessous. Chacun est tenu de présenter oralement une partie de l’application et de savoir expliquer tout le code même celui qu'il n a pas écrit. ***Je me réserve le droit de différencier la cote en cas de différence trop flagrante de travail.***

***Toutes les étapes de ce dossier doivent être conservées. Il est obligatoire d’avoir un répertoire par étape***.

***Vous êtes obligés de suivre la séquence des étapes.***

## Échéances

L’évaluation sera une évaluation continue qui se pratiquera chaque semaine dans les laboratoires **selon la pondération ci-dessous.** Voici les échéances à respecter

|  |  |  |  |
| --- | --- | --- | --- |
| Point | Date (semaine du) | TELECOM GESTION Pondération | Indus Pondération |
| Etape 0 | 24 janvier | 1,5 | 2 |
| Etape 1 | 31 janvier | 1,5 | 2 |
| Etape 2 | 7 février | 2 | 2 |
| Etape 3 | 14 février | 1,5 | 2 |
| Etape 4 | 21 février | 1,5 | 2 |
| Etape 4 Bis | 28 février \* 1 mars | 3 | 4 |
| Etape 5 | 14 mars | 2 | 2 |
| Etape 6 | 21 mars | 3 | 2 |
| Etape 7 | 28 mars | 2 | 1 |
| Etape 8 | 4 avril | 2 | 1 |

**Les étudiants seuls ont droit à 25 % de points en plus**

Pas plus de deux points sont évalués par semaine. L’évaluation se terminera à la dernière séance de laboratoire du quart de l'année.

**En seconde session, c’est la même application il faut simplement terminer les points qui n’ont pas été faits correctement.**