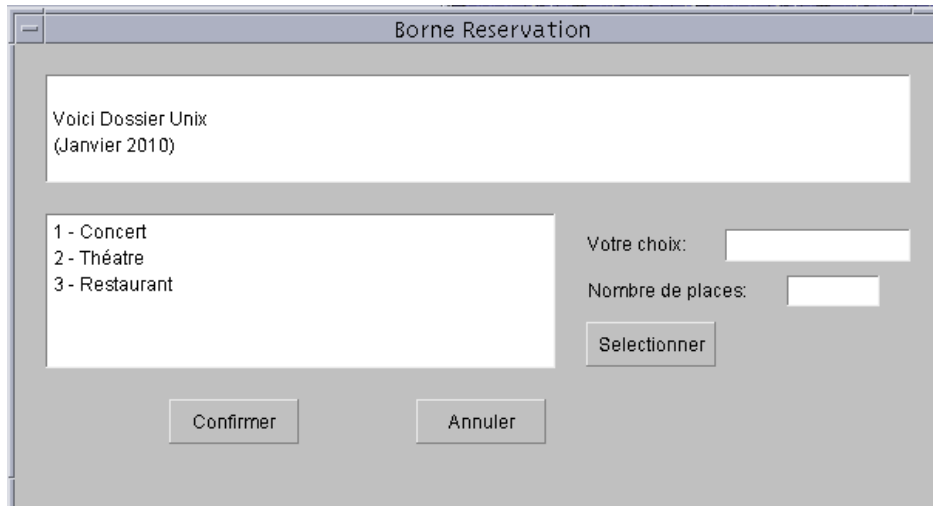


On demande de simuler des bornes d'informations que l'on peut trouver à l'entrée de certains lieux assez fréquentés.

Ces bornes affichent des publicités en continu mais permettent surtout d'obtenir des informations culturelles et d'effectuer certaines réservations (place(s) de concert, de théâtre, table de restaurant, ...)

Dans le cas de notre application, on se limitera uniquement aux concerts, théâtres ou restaurants.



Pour cela, on dispose d'une application Serveur, et d'une application Borne. Celle-ci communiquera avec le Serveur par une file de messages.

Le processus Serveur s'occupe de tout, c.à.d.

Il crée tout ce dont l'application a besoin : la file de message, les sémaphores, la mémoire partagée et le processus Publicite qui transmettra la publicité qui sera affichée sur la Borne (cela se fera par mémoire partagée).

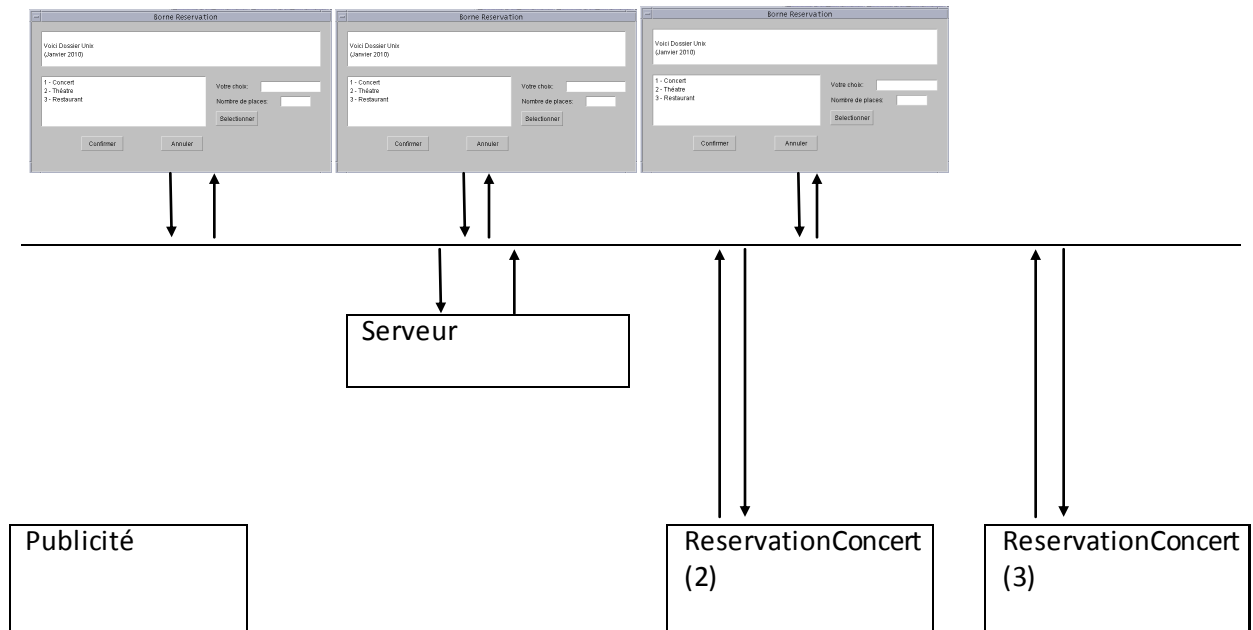
Ensuite, le processus Serveur attend une requête, la détermine et lance éventuellement le processus ReservationXXX (ReservationConcert, ReservationTheatre, ReservationRestaurant) correspondant, ensuite il se remet en attente de la requête suivante.

Une fois le processus ReservationXXX lancé, celui-ci communiquera avec la Borne tout le temps de la transaction.

Une fois la transaction terminée, le processus ReservationXXX se termine et doit être enlevé de la table des processus.

---

Exemple : Trois Borne sont lancées, la première attend une transaction, la seconde et la troisième ont effectué une réservation de places de concert.



## Le processus **Borne** :

Le processus **Borne** affiche une publicité (c.-à.-d. la chaîne de caractères lue dans la mémoire partagée) et cela en continu et sans intervention.

Un passant peut sélectionner un concert, un théâtre ou un restaurant afin de consulter la liste des disponibilités pour éventuellement faire une réservation.

En cas de réservation, il indiquera son choix ensuite le nombre de places souhaitées.

Un message sera alors reçu pour lui indiquer si son choix est possible ou non (il reste trop peu de places libres). Il devra confirmer son choix ou l'annuler.

Pour cela, il effectue une requête au Serveur par file de messages et attend la réponse.

La requête est :

Une consultation :      requête = **CONSULT**.  
Donnée = le choix effectué.

Un affichage des différentes possibilités apparaît.

(ex : liste des concerts prévus avec le nombre de places libres, le lieu ...)

Le processus Borne communiquera ensuite avec le processus ReservationXXX correspondant.

Il devra confirmer sa réservation ou l'annuler.

La requête sera alors      requête = RESERVATION  
Donnée = (Choix, Nombre de places)  
requête = CONFIRMER  
Donnée = /  
requête = ANNULER  
Donnée = /

Dans tous les cas, si pas de réaction de la part du passant, la demande est annulée après 20 secondes et l'écran réaffiche la situation de départ.

## Le processus **Serveur** :

Le processus **Serveur** travaille selon le principe d'un démon.

Il crée toutes les ressources nécessaires à l'application. (le fichier Trace.log, les files de messages, la mémoire partagée, les sémaphores)

Lance le processus **Publicite**.

Il attend une requête.

A chaque requête, il lance un processus **ReservationXXX** et se remet de suite en attente de la requête suivante.

A la fin de l'application, il supprime toutes les ressources.

Le Serveur doit connaître tous les processus **Borne** car le processus **Publicite** doit communiquer à toutes les bornes la nouvelle pub.

## Le processus **ReservationXXX** : (ReservationConcert,ReservationRestaurant,ReservationTheatre)

Le processus **ReservationXXX** traite entièrement la requête, et ne se termine qu'une fois celle-ci clôturée.

Il y a donc autant de processus **ReservationXXX** que de passant ayant effectué une requête.

La réponse de **Reservation** est envoyée directement au processus Borne.

Si requête = <b>CONSULT</b>	<b>ReservationXXX</b> lit dans le fichier correspondant et transfère les données au Client.
Si requête = <b>RESERV</b>	Il attend une réponse.( soit RESERVATION, soit ANNULE) <b>ReservationXXX</b> effectue le retrait du nombre de place(s). Ce même processus renvoie au Client une requête CONFIRMATION ou ANNULE (dans le cas où il n'y a plus assez de places). Le processus <b>ReservationXXX</b> se termine.
Si requête = <b>ANNULER</b>	<b>ReservationXXX</b> se termine.
Si requête = CONFIRMATION	Le processus exécute le retrait.

**Remarque** : Lorsque le processus **ReservationXXX** se termine, il doit être supprimé de la table des processus.

Un début de programmation est proposé dans le répertoire habituel.  
(pour rappel : /export/home/public/Merced/Dossier2\_2010\_2011 sur la machine sunray1v440).

## Le processus **Publicite** :

Le processus **Publicite** lit dans le fichier Publicite.dat des enregistrements.  
Ceux-ci seront écrits dans un fichier texte ordinaire, une première ligne pour la pub, une 2<sup>ème</sup> ligne indiquant le nombre de secondes que celle-ci sera affichée.

Ce processus écrira *szPub* dans la mémoire partagée, émettra un signal SIGUSR1 à tous les processus **Borne**, et attendra *Secondes* secondes.  
Quand tous les enregistrements sont lus, il recommence au début, et ainsi de suite.

## La solution comprend plusieurs étapes.

(Conseil : Réalisez-les dans l'ordre proposé)

- 1- Réservation des places.

En fait, il faut faire une gestion simplifiée des fichiers Concert.dat, Theatre.dat et Restaurant.dat. (La gestion du fichier est libre). Mais comme il y a plusieurs processus ReservationXXX, elle devra être faite par des appels de fonctions de bas niveau.

Pour simplifier au maximum, le fichier est constitué d'un maximum de 10 éléments.

## 2- Cas de plusieurs Bornes :

Il est évident que l'application sera testée avec plusieurs processus **Borne**.

Il faut donc associer le **Borne** au processus **ReservationXXX** correspondant.  
 Pour cela, dans le Serveur, lorsqu'une requête **CONSULT** arrive, il faut faire correspondre (dans une table), le processus **Borne** avec le processus **ReservationXXX** lancé. Ce qui permettra de transmettre les requêtes suivantes de la Borne au bon processus ReservationXXX.

De plus, les Borne devront afficher les publicités.

Une méthode simple sera de faire appartenir toutes les bornes au même groupe de processus.  
 Pour cela, lorsque la première Borne se connecte, elle crée son propre groupe, et toutes les autres Borne s'y attacheront. Ce qui est possible, car le Serveur mémorise le groupe de la première Borne et la transmet aux futures Bornes.

(Cela marche, il suffit de lancer les Bornes sur le même terminal et en arrière plan)

## 3- Cas des processus zombies :

En fait, quand le processus **ReservationXXX** se termine, il occupe encore une entrée dans la table des processus.

Pour s'en convaincre, tapez la commande

```
ps -ael | grep $USER | grep -v \ /opt | grep -v \ /usr | grep -v dtgreet
```

Il faut les supprimer.

(un fils, lorsqu'il se termine, émet un signal SIGCHLD à son père)

## 4- Cas de plusieurs Client (bis) :

Si 2 processus **Borne** réservent les mêmes places, il faut vérifier qu'elles ne seront vendues qu'une seule fois, et qu'il en reste assez.

(Donc que *PLibre* >= 0)

(placer donc des lock de fichier sur l'enregistrement en question).

## 5- Modification des fichiers .dat :

Ecrire un processus **GestServ**.

En fait, le processus **GestServ** s'occupe de la gestion des fichiers \*.dat

Par exemple, un concert est annulé ,ajout d'un autre concert, ajout de places.

On ne peut pas le modifier si quelqu'un travaille dessus.

Il faut donc attendre que tous les processus **ReservationXXX** soient terminés, et interdire de nouvelle connexion.

Ceci se fait par l'usage de sémaphores (il ne manquait plus qu'eux).

On admet un maximum de 15 processus ReservationXXX.

Un sémaphore qui servira de compteur de **Borne**. (On incrémente de 1 lorsqu'un passant lance une requête)

Et décrémente de 1 lorsque celle-ci se termine, Il suffit alors d'attendre que ce compteur passe à 0 pour modifier les fichiers.

Un autre sémaphore pour interdire de nouvelle requête.

Une fois les modifications du fichier faites, il faut prévenir les différents processus Borne par un signal pour que celui-ci puisse tenir compte de ces modifications.

De plus, il faut être une personne privilégiée pour utiliser ce programme.  
Il sera donc protégé par un mot de passe.

6- Reste le fichier des Publicites.

Si l'administrateur ajoute, supprime ou modifie des publicités, (rappelons qu'il s'agit d'un fichier texte), il faut prévenir le processus **Publicite** de la modification.  
(pour simplifier on se contente de maximum 10 publicités différentes)

Le travail se fait par équipe de 2 étudiants et sera défendu en janvier 2011.

Il sera testé sur la machine de l'institut. (**sunray1v440**).

Un des professeurs responsables peut questionner l'équipe.