Projet Système – Agence de tourisme

François HAURIT, Charaf SALMI, Sophie SURMONT, Groupe 204

Sommaire

[Les processus développés 3](#_Toc221678315)

[Processus pAccueil 3](#_Toc221678316)

[Processus pGuichet 4](#_Toc221678317)

[Processus pAdministration 4](#_Toc221678318)

[Processus pDirection 4](#_Toc221678319)

[Les fichiers utilisés 4](#_Toc221678320)

[Les signaux 5](#_Toc221678321)

[Les tubes 5](#_Toc221678322)

[Schéma de l’interaction des processus 5](#_Toc221678323)

[Méthode de travail 5](#_Toc221678324)

[Difficultés 6](#_Toc221678325)

[Satisfactions 6](#_Toc221678326)

[Détails techniques 6](#_Toc221678327)

[En conclusion 6](#_Toc221678328)

# Les processus développés

Le programme se compose en différents processus s’exécutant les uns pendant les autres, c'est-à-dire en parallèle. Ces processus représentent chacun un service de ce que se veut représenter le projet, « une agence ».

## Processus pAccueil

* Lit les transactions dans le fichier fTransactions\_clients
  + La transaction est une structure
  + Si transaction.code = C alors écrire le nombre de place disponibles ou aucune sur le terminal et dans le journal
  + Sinon, pour chaque transaction, vérifier l’existence du voyage dans fVoyage
    - S’il n’existe pas
      * C’est la première demande
        + Créer [voy].desc dans le répertoire « Acreer » qui contiendra le nombre de transactions qui ont demandé ce voyage (minimum)
        + Créer [voy].fa (file attente) dans le répertoire « Reservation » et y mettre cette transaction
      * C’est une demande de voyage qui n’a pas été référencée
        + Incrémenter le fichier .desc
        + Ajouter cette transaction dans .fa
    - S’il existe
      * Ecrire la transaction dans le tube
  + Si, durant le traitement, on reçoit un signal de la part de pAdministration de création de nouveau voyage, on envoi le contenu du voyage correspondant
* S’il reçoit un signal de pAdministration, il lit le tube tAdmin-accueil
* Envoi les structures transaction à pGuichet par le tube tAccu-guichet
* S’arrête, se met en pause ou redémarre à la demande de pDirection via le tube fTransactions\_clients

## Processus pGuichet

* Se met en pause lorsque le tube tAccu-guichet est vide
* S’arrête lorsque pAccueil et ce tube sont fermés

Il y a plusieurs instances de pGuichet, chacune agît parallèlement. Il y a un tube (taccu-guichet) commun à tous.

Ce tube permet l’envoi et la réception de transactions entre processus. Il y a deux types de transactions :

* Effectuer une réservation
  + On vérifie les places disponibles
  + Si suffisamment de places
    - On met à jour le fichier fVoyage
    - On ajoute un article à Voy.reserv
  + Sinon
    - Un article est ajouté à Voy.fa
* Annuler une réservation
  + Supprimer l’article de Voy.reserv
  + Mettre à jour le nombre de places dans fVoyages
  + Faire passer les éléments de Voy.fa en tête de fTransactions\_clients à concurrence du nombre de places libérées

## Processus pAdministration

* Lit fTransactions\_Admin, ce fichier est composé d’enregistrement créé par le programme create, la structure des enregistrements est de type Transaction\_admin. Donc, le nom du voyage, le code, qui est C (création) ou F (suppression), ce programme sert donc à traité ce fichier, il supprime et créer des enregistrements dans le fichier fVoyages, et le créer si nécessaire.
* Ce processus gère les signaux, il recevra un signal lui disant de s’arreter, il sort de sa boucle de lecture et ferme tout ses fichiers.
* Création :
* Quand un nouveau voyage est créé, il rajoute un élément au fichier fVoyages, et écrit ce même enregistrement dans un tube le liant à pAccueil. Il informe aussi pAccueil de cette ajout.
* Quand il supprime, il tasse les données du fichier, et remplace le dernier article de l’article par un blanc, à la création d’un nouvelle article, il testera si le dernier est un blanc ou pas, et le remplacera (ou pas), pour avoir un fichier toujours bon.

## Processus pDirection

* Lance pAccueil et pAdministration.
* Attends la mort de pAccueil pour ordonner à pAdministration de mourir.

# Les fichiers utilisés

* fVoyages : composé d’enregistrements de type Produit (voir type\_definitions.h) renseigne sur le remplissage des places du voyage concerné
* *Reservations/Voy*.reserv : composé des réservations acceptées pour le voyage « *Voy* »
* *Reservations/Voy*.fa : composé des réservations non satisfaites pour le voyage « *Voy* », dès qu’un client libère sa réservation, les éléments présents sont transférés vers « fTansactions\_clients », dans le but d’être retraités par pAccueil.
* fTansactions\_clients

# Les signaux

La communication interprocessus se fait à l’aide de signaux. Ils sont envoyés de :

* pDirection à pAccueil et pAdministration
* pAdministration à pAccueil
* pAccueil à pGuichet

# Les tubes

Les tubes servent à l’échange de données interprocessus. Ils y en a entre :

* pAdministration et pAccueil
* pAccueil et pGuichet

# Schéma de l’interaction des processus

Les flèches représentent des signaux.

# Méthode de travail

La réalisation de se projet devait requérir sept semaines, mais en réalité, une de plus n’aurait pas été superflue.

## Difficultés

La plus grande difficulté rencontrée a été la compréhension du sujet en lui-même. Non pas que la technique était effrayante, mais le travail demandé n’était pas clair, les termes utilisés mélangeaient avec subtilité images et détails techniques sans qu’il ne nous ait été possible d’en dégager une certaine logique ou structure dans le développement. La même information avait plusieurs dénominations à travers tout l’énoncé. Pour comprendre ce que devait réaliser pAccueil ou pAdministration par exemple, il nous a fallu relire minutieusement chaque paragraphe plusieurs fois en tentant de faire des liens un petit peu partout, les spécifications étant non pas décrites mais éparpillées sur toutes les pages, se répétant, se complétant ou se faisant moins précises tout au long du sujet. Les titres ne nous ont pas semblés clairs. Plus que du travail d’analyste-programmeur technicien système, il nous a fallu un travail d’interprète.

Ce long et complexe travail d’analyse a ensuite débouché sur l’implémentation des algorithmes extraits, en C, sous linux, en compilant avec GCC. Au niveau technique, nous savions utiliser les *tubes*, les *fichiers*, les *exec*, mais les signaux nous ont donné un petit peu de piquant.

Nous ne pensons pas avoir fini d’implémenter les algorithmes.

## Satisfactions

Oui, ce sujet n’a pas été qu’un long déboire. Nous ne sommes pas mécontents d’avoir vu ce qu’il pouvait se faire de difficile à comprendre en termes de spécifications logicielles. Nous avons relativisé l’image que nous avions de la faculté de nos futurs clients dans le monde du travail à nous expliquer ce qu’ils attendent de nous.

Nous sommes aussi rassurés d’avoir pu mettre en pratique concrète la théorie acquise dans le cours et reprendre l’exercice de ce langage de programmation abandonné après la S1. Quoi qu’on en dise, la programmation en C est élégante et demande une rigueur qui nous tient à cœur. Notre trinôme a en commun le fait de l’apprécier particulièrement.

## Détails techniques

Pour aller plus vite, nous avons utilisé une subversion, nous permettant de partager nos sources et de travailler de façon collaborative sans perdre de temps en futilités organisationnelles (échange de fichiers, synchronisations, gestion de versions…). L’adresse du dépôt est la suivante : <http://code.google.com/p/pps-projet-systeme/>

# En conclusion

Il est dommage que nous fussions dans un contexte de surcharge de travail, avec plusieurs autres projets à traiter en parallèle. Nous aurions aimé pouvoir travailler dans les mêmes conditions que lors de nos derniers travaux dirigés notés.