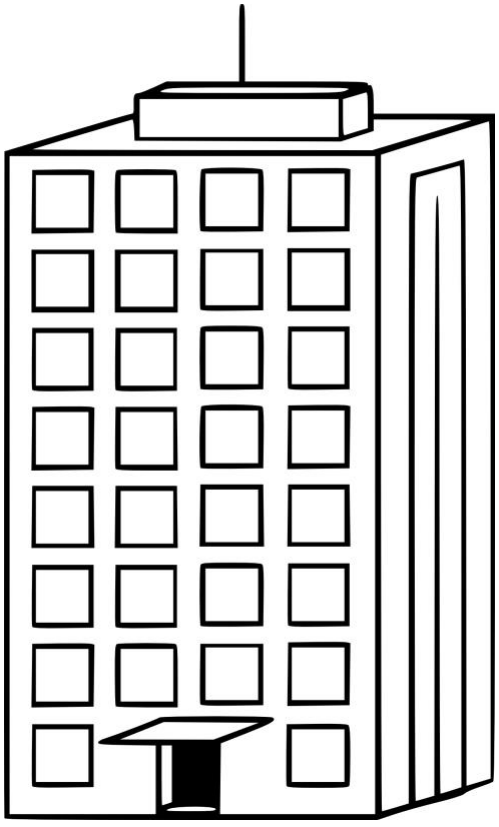
LO41 : Systèmes d'exploitation

Projet de programmation

« Environnement d'un Immeuble »



Réalisé par :

Thomas Contini
Shunli Feng

Encadré par :

Philippe Descamps
Renan Zeo

Table des matières

TABLE DES MATIERES **2**
INTRODUCTION ET CONTEXTE **3**
INTRODUCTION 3
MOTIVATIONS 3
CONTEXTE 4
COMPREHENSION DU SUJET **5**
IDENTIFICATION DES ELEMENTS CLES 5
EXEMPLE DE SCENARIO 5
SCHEMA DES COMMUNICATIONS POTENTIELLES 6
RESEAU DE PETRI 7
CHOIX D'IMPLEMENTATION **8**
STRUCTURE DU PROGRAMME 8
UTILISATION DES OBJETS IPC 8
SCHEMA DES PRINCIPAUX MOYENS DE COMMUNICATION 9
CONCLUSION **10**
PROBLEMES RENCONTRES 10
PISTES D'AMELIORATION 10

Introduction et Contexte

Introduction

Dans le cadre de l'UV d'informatique LO41 nommée « systèmes d'exploitation », nous devions réaliser un projet de programmation à l'aide des différents outils systèmes évoqués tout au long de l'année en cours, TD et TP. Ce projet permet principalement la mise en pratique des acquis obtenus tout au long du semestre. Le programme devra être développé dans une logique de portabilité maximum. Il devrait pouvoir s'adapter et fonctionner sur la plupart des environnements systèmes pour maximiser la compatibilité.

Les outils à notre disposition nous ont été introduit tout au long du semestre et sont les suivants :

- Les processus
- Les tubes
- Les signaux
- Les files de message
- Les mémoires partagées
- Les sémaphores
- Les moniteurs
- Les threads

Il sera également intéressant de fournir différentes représentations graphiques comme un réseau de Pétri pour simplifier la compréhension du projet.

Motivations

L'IA se faufile dans notre vie quotidienne. Elle opère en ingérant de nombreuses données qui par leurs analyses peuvent nous conduire à envisager de nouveaux champs d'innovation. L'un de ces champs s'est ouvert en direction de l'IOT qui nous aide notamment à comprendre comment nos appareils fonctionnent, comment ils sont utilisés, et quels types de problèmes ils peuvent rencontrer.

Les ascenseurs n'échappent pas à cette révolution. Intelligence embarquée, capteurs, services dédies consolident les solutions que proposent de nombreux constructeurs dans cet écosystème.

Par exemple, si un ascenseur est bloqué dans un immeuble, le technicien le plus proche qui dispose des bons outils et des pièces nécessaires sera automatiquement identifié et mobilisé pour intervenir sur le champ. En sus, les données de fonctionnement seront communiquées au technicien pour sa tournée de révisions. Il disposera ainsi des informations les plus récentes et les plus pertinentes pour adapter sa check-list en approfondissant certains points qui pourraient devenir critiques. L'ascenseur de demain se veut connecter.

Contexte

Dans le sujet proposé, il est question de réaliser un programme permettant de simuler l'environnement d'un immeuble d'habitation et/ou de bureau, à l'aide des outils systèmes tels que les processus, threads, objets IPC, etc...

Dans notre cas d'étude, nous traiterons des cas suivants :

- Notre immeuble comporte 25 étages et 3 ascenseurs pour desservir ces derniers.
- Un certain nombre de résidents sont déclarés comme habitants de l'immeuble.
- Un visiteur/livreur peut être amené à échanger avec un résident de l'immeuble qui pourra lui faciliter l'accès à son logement.
- Lorsqu'un visiteur arrive et s'identifie dans le hall d'accueil, son contact dans l'immeuble est prévenu. Dès que l'autorisation est donnée par l'hôte, le chemin est indiqué au visiteur. On lui spécifiera l'ascenseur à prendre.
- Dans un objectif d'optimisation et d'économie d'énergie, une « intelligence artificielle » regroupe les passagers dans les ascenseurs en fonction de leur destination, afin qu'ils arrivent plus vite à l'étage souhaité, avec moins d'arrêts.
- Chaque ascenseur a bien entendu une capacité limitée.

Les cas suivants n'ont pas été considéré comme des priorités. Ils pourront cependant être étudiés en complément :

- *SI l'ascenseur est bloqué dans un immeuble, le technicien le plus proche qui dispose des bons outils et des pièces nécessaires sera automatiquement identifié et mobilisé pour intervenir sur le champ.*
- *Certains accès sont limités...*

Compréhension du sujet

Identification des éléments clés

Après consultation du sujet, il se dégage différents éléments centraux permettant de définir les premiers objectifs. En autre, nous avons identifié les cinq principales instances qui devront être développées, listées ci-dessous :

- L'immeuble et son système d'accès,
- L'ascenseur,
- Le visiteur/livreur,
- Le résident,
- Le technicien.

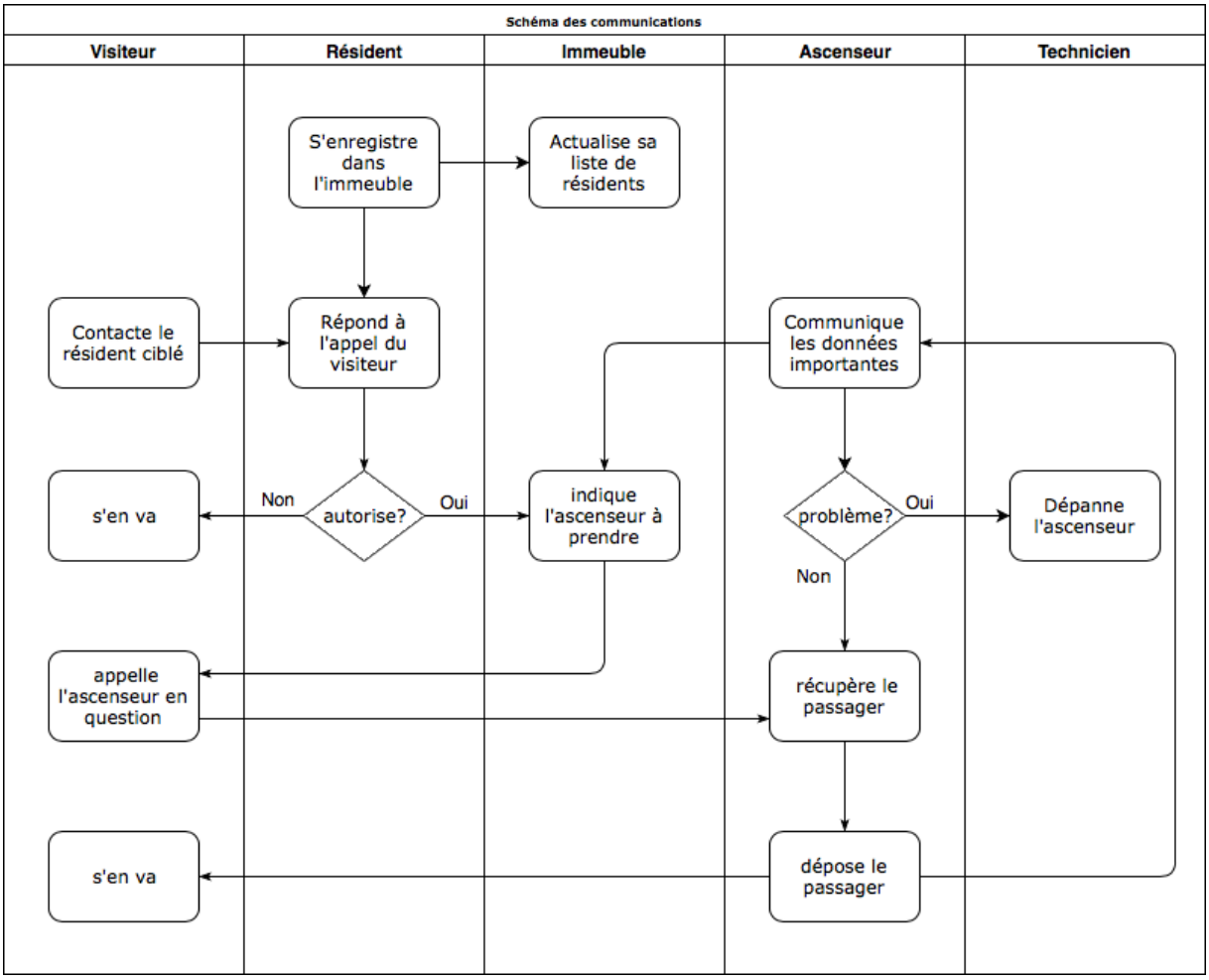
Exemple de scénario

A partir de cela, nous avons donc imaginé différents scénarii de mise en œuvre pour identifier les différentes phases. En voici un exemple :

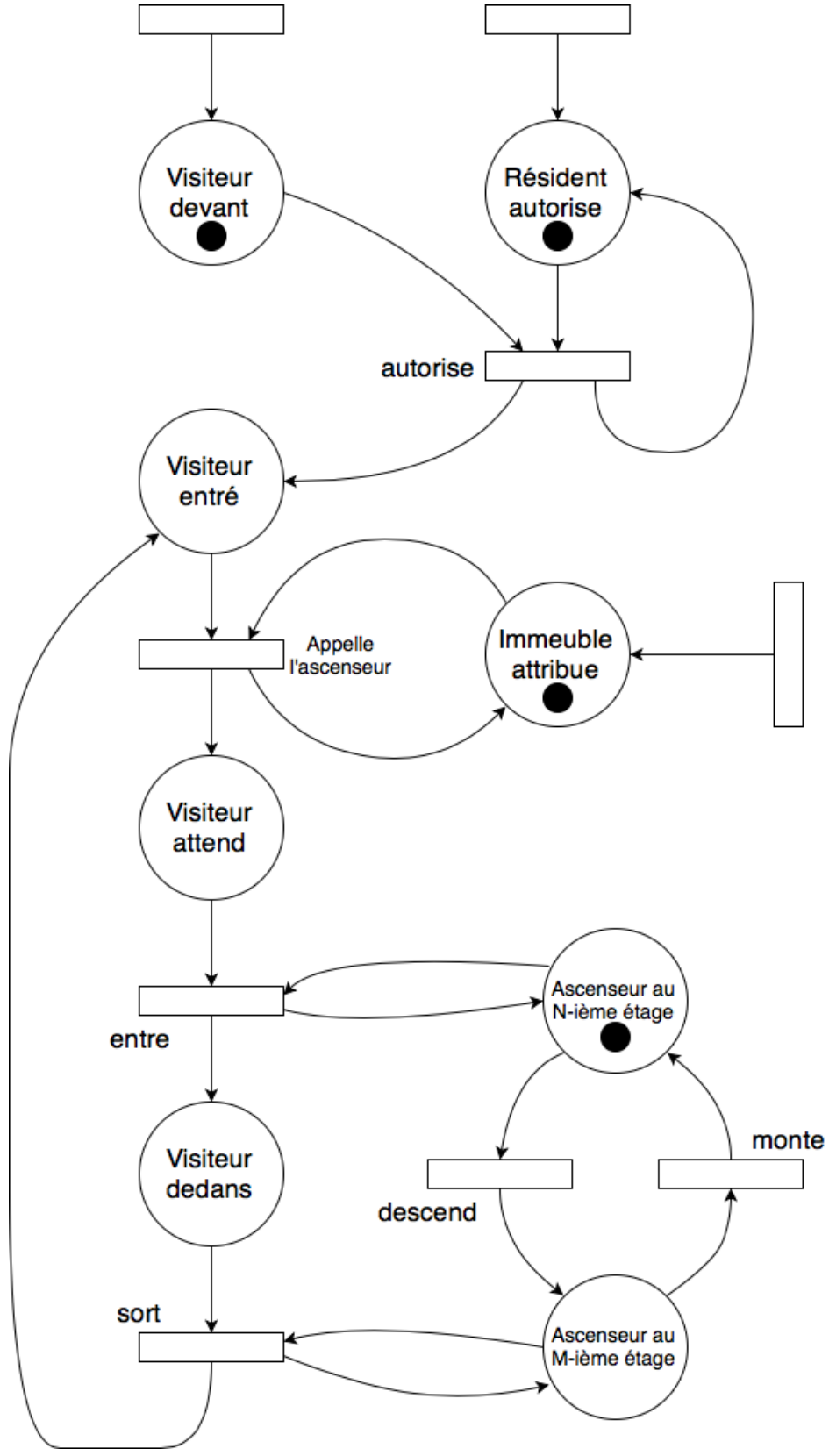
1. Lorsqu'un visiteur se présente dans le hall d'accueil de l'immeuble, il sait d'avance la destination qu'il souhaite atteindre. On peut imaginer pour cela que le visiteur a accès à une liste des résidents avec leur numéro/identifiant d'appartement spécifié.
2. Via l'interface proposée, le visiteur contacte le résident en question en sélectionnant son numéro d'appartement (qui pourrait être une composition de l'étage et du numéro de porte).
3. Si le résident n'est pas présent chez lui, le visiteur ne reçoit aucun retour et s'en va.
4. Sinon, le résident reçoit l'appel et décide d'autoriser l'accès ou non.
5. Si l'autorisation est donnée, l'interface fournira le numéro d'ascenseur à prendre au visiteur. Ce numéro aura été choisi de façon intelligente, par rapport à l'utilisation actuelle des ascenseurs.
6. Le visiteur appelle alors l'ascenseur et y entre une fois que ce dernier est arrivé.
7. Lorsque l'ascenseur atteint l'étage visé par le visiteur, il l'avertit et le laisse descendre.

On comprend aisément de part ce scénario très simplifié que l'enjeu principal du projet sera le dialogue entre nos différents éléments. L'immeuble, les ascenseurs, les visiteurs ainsi que les résidents devront être en mesure de communiquer efficacement et à tout moment.

Schéma des communications potentielles



Réseau de Pétri



Choix d'implémentation

Structure du programme

Nous avons fait le choix d'utiliser les processus pour modéliser les différents éléments du projet. Nous aurons donc un processus par instance identifiée (immeuble, ascenseur, visiteur, résident et technicien). Ces derniers devront être en mesure de communiquer entre eux de façon simple et explicite. Ils échangeront à l'aide de plusieurs objets IPC décrits dans la partie suivante.

Avant de commencer à programmer quoi que ce soit, nous avons voulu répartir les différentes tâches que devront réaliser chaque processus.

- **Immeuble :**
  - Stocke l'identifiant et le lieu où habite chaque résident déclaré.
  - Doit être en mesure de le communiquer aux visiteurs qui veulent entrer.
  - Doit récupérer l'identifiant de chaque ascenseur ainsi que les informations utiles du type 'Nombre de places restantes' et 'étage actuel'.
  - Sait diriger intelligemment les visiteurs vers l'ascenseur approprié.
- **Ascenseur :**
  - Reçoit les appels des visiteurs qui appellent l'ascenseur.
  - Transporte les visiteurs d'un étage à un autre et les avertit de leur arrivée.
  - Se déplace intelligemment pour limiter les aller-retours.
- **Résident :**
  - S'enregistre auprès de l'immeuble.
  - Donne ou non l'accès à un visiteur qui le contacte.
- **Visiteur :**
  - Contacte un résident pour demander l'accès, à l'aide de la liste des habitants fournie par l'immeuble.
  - Appelle l'ascenseur, se rend au bon étage, puis repart au bout d'un certain temps.

Utilisation des objets IPC

Il a été convenu d'utiliser différents objets IPC pour simplifier la communication entre les processus décrits précédemment. On se servira donc de files de message, de mémoires partagées et de sémaphores.

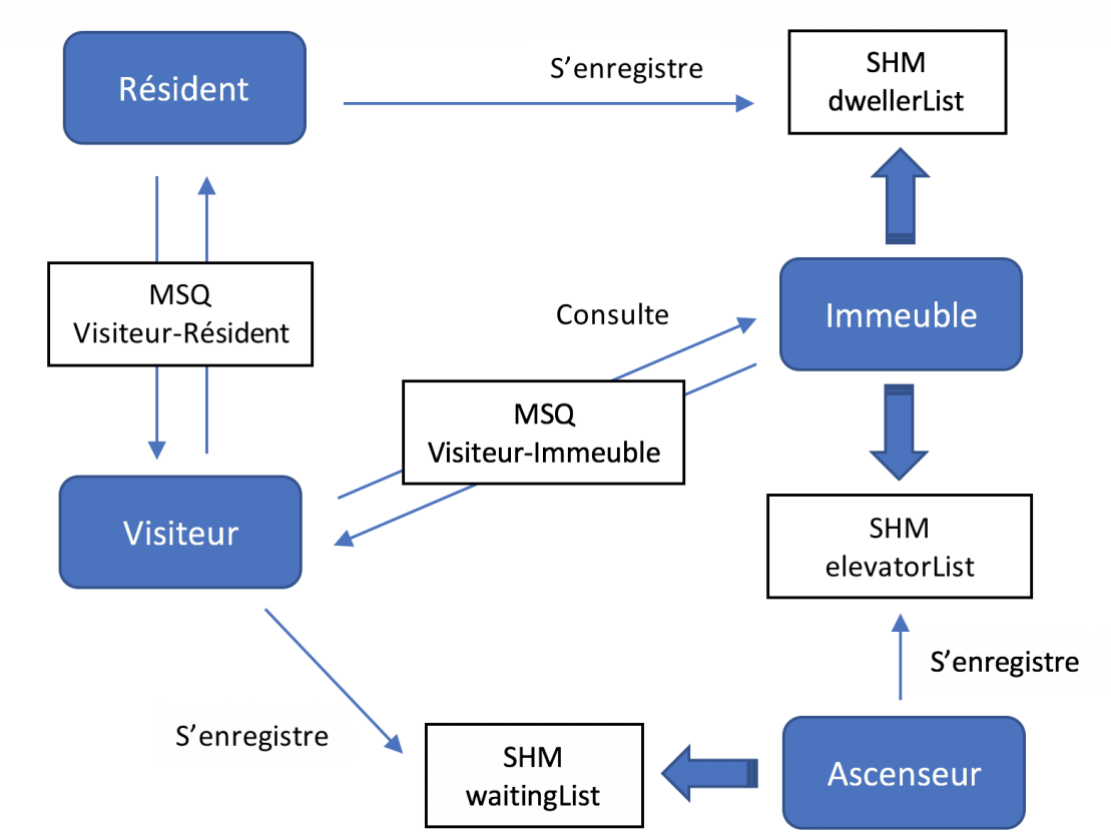
- **Files de message :**
  - Visiteur-Résident alias msqVR : file utilisée pour qu'un visiteur communique avec un résident.
  - Visiteur-Immeuble alias msqVI : file permettant au visiteur de demander quel ascenseur prendre auprès de l'immeuble.

- **Mémoires partagées :**
  - 'DwellerList' alias shmDL : liste de tous les résidents de l'immeuble avec leur pid, étage et porte.
  - 'ElevatorList' alias shmEL : liste des ascenseurs, contenant leur pid, étage actuel ainsi que leurs places restantes (pour des questions de simplicité, le pid de l'immeuble y sera aussi enregistré).
  - WaitingList' alias shmWL : liste contenant le pid, l'étage actuel et l'étage de destination de chaque visiteur ayant appelé l'ascenseur.
- **Sémaphores :**
  - semDL : sémaphore protégeant l'accès à la mémoire partagée 'DwellerList'.
  - semWL : sémaphore protégeant la 'WaitingList'

Pour toute synchronisation ne nécessitant pas de transfert de variables, nous utiliserons les signaux système.

- *SIGINT* sera redéfini pour permettre la destruction automatique des objets IPC créés par un processus avant que celui-ci meurt.
- *SIGUSR1* et *SIGUSR2* seront également définis pour permettre certaines actions comme l'actualisation visuelle des listes.

Schéma des principaux moyens de communication



Conclusion

Problèmes rencontrés

Pendant la réalisation de ce projet, nous avons fait face à différents problèmes d'implémentation qui nous ont empêché de mener à bien tous les objectifs que nous nous étions fixés.

Le principal problème a été le stockage de données dans les mémoires partagées. En effet, ces espaces mémoires sont utilisés pour stocker l'équivalent de tableaux à deux dimensions. Cette perspective n'avait jamais été abordé en TP auparavant et il a fallu s'armer de patience pour réussir à écrire et lire de manière simple dans ces zones mémoires.

Pistes d'amélioration

Pour conclure ce rapport, l'ensemble des éléments ci-dessus qui ont été programmés fonctionnent parfaitement. Il nous reste cependant plusieurs pistes d'amélioration.

Tout d'abord, nous n'avons pas implémenté le technicien dans la version actuelle du projet. Nous y avons tout de même réfléchi et le code de l'ascenseur est déjà adapté pour un possible ajout de cette fonctionnalité.

De plus, certaines choses restent à améliorer. En effet, à l'heure actuelle, seuls les visiteurs peuvent se déplacer dans l'immeuble. Il serait intéressant de permettre aux résidents de pouvoir entrer et sortir du bâtiment, ou même de rendre visite à un de leur voisin. Les intelligences artificielles de l'immeuble et des ascenseurs restent très peu développées. Nous avons préféré nous concentrer sur la partie système (notamment la communication interprocessus). Il serait possible par exemple de considérer d'avantages de paramètres dans le choix de l'ascenseur à attribuer à un visiteur (cibles de l'ascenseur, direction…).

Pour finir, le code actuel s'exécute via des lignes de commande à saisir nous-même dans différents terminaux. Il serait intelligent de regrouper plusieurs processus sous un même terminal (avec des forks et un système de Prompt pour savoir qui parle) ou encore d'automatiser la phase de lancement.