

Ecole Publique d'Ingénieurs en 3 ans

Rapport

PROJET SIMULATION ÉPIDÉMIE

Disney Town

le 8 janvier 2021,

Sarah Brood,
sarah.brood@ecole.ensicaen.fr

Thomas Savignac,
thomas.savignac@ecole.ensicaen.fr



www.ensicaen.fr

TABLE DES MATIÈRES

GESTION DES CITOYENS	2
GESTION GÉNÉRALE, LES CAS PRÉCIS	2
GESTION DES THREADS CITOYEN	2
GESTION DES TOURS	2
TIMER ET SES ALARMES, PIPES	2
FIN DU PROGRAMME	2
GESTION DE L'ÉPIDÉMIE	3
MÉMOIRE PARTAGÉE	3
NOTIFICATION DES TOURS	3
AFFICHAGE DU GRAPHIQUE	3
INTERFACE HOMME MACHINE	4
CRÉATION ET LANCEMENT DE PROGRAMMES	5
INTÉGRATIONS EXTÉRIEURS	5

INTRODUCTION

Ce projet est le dernier travail pratique de la matière Systèmes d'Exploitations. Le but de celui-ci est de mettre en œuvre nos connaissances pour créer une simulation d'épidémie. Pour cela, nous avons utilisé différents mécanismes vu en cours.

IMPLÉMENTATION

Gestion des citoyens

Gestion générale, les cas précis

La gestion générale des citoyens se fait dans le programme *citizen_manager*, on peut y retrouver toutes les conditions de mouvement, les calculs de transmission, et les actions des citoyens spéciaux tout cela en fonction des différents types de cases sur la carte.

La lecture et l'extraction des différentes informations dans le sujet furent assez compliquées, car le nombre de conditions était assez grand. De ce fait, nous avons opté pour l'utilisation d'un tableur pour noter les différences de chaque type de personne à partir du comportement d'un simple citoyen.

Gestion des threads citoyen

Le programme *citizen_manager* exécute à chaque tour de jeu le comportement de 37 citoyens, chaque citoyen a son propre thread. Comme chaque citoyen modifie potentiellement la ville et les citoyens autour de lui de part ses actions, nous avons opté pour l'utilisation de sémaphores pour l'accès à ces données.

Nous avons utilisé le code de l'exemplier de Monsieur Lebret pour les sémaphores.

Gestion des tours

timer et ses alarmes, pipes

Chaque tour doit s'exécuter avec un intervalle de plusieurs secondes, pour se faire, nous avons utilisé la gestion d'alarme pour les temporisations. Pour notifier la fin des tours, on a implémenté des tubes nommés pouvant ainsi communiquer avec le gestionnaire de l'épidémie.

Nous avons utilisé le code de l'exemplier de Monsieur Lebret pour les tubes nommés.

Fin du programme

Afin d'arrêter tous les processus à la fin des 100 tours, le timer envoie un message différent du message habituel, défini avec grande originalité "end" qui sera transmis à travers tous les programmes afin de définir leur fin.

Gestion de l'épidémie

Mémoire partagée

Les programmes *citizen_manager* et *epidemic_simulation* utilisent tous les deux la ville, nous avons donc utilisé le système de mémoire partagée. C'est *epidemic_simulation* qui crée cette partie de mémoire, *citizen_manager* récupère ensuite celle-ci.

Nous avons eu des problèmes avec la gestion de cette mémoire partagée, car en premier lieu les programmes étaient fermés avec une simple interruption clavier ce qui ne permettait pas de délier la mémoire partagée. Cela gênait les nouvelles simulations, nous avons donc créé un script bash qui permet de nettoyer les possibles résidus des dernières simulations avant le lancement d'une nouvelle simulation.

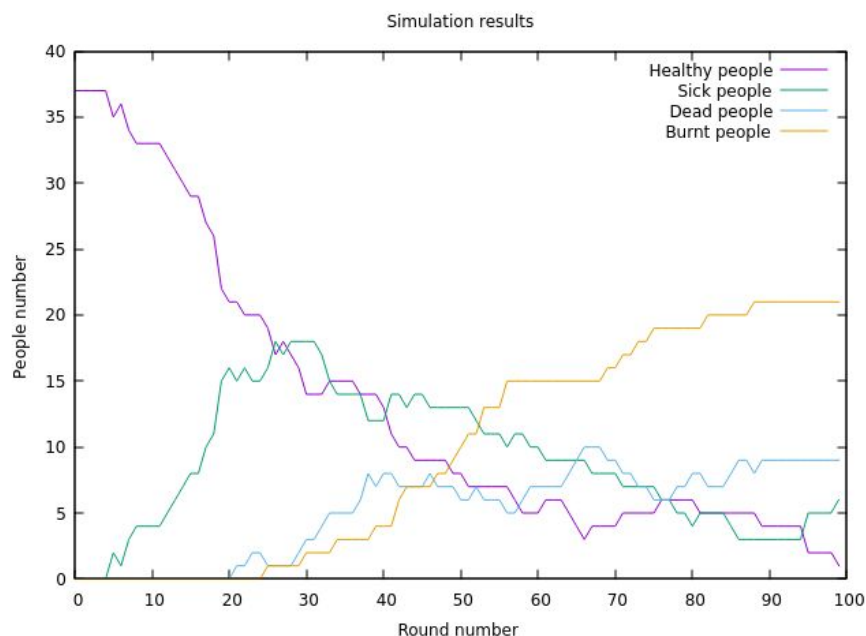
Notification des tours

Pour pouvoir notifier le programme *citizen_manager*, on a créé un tube nommé entre celui-ci et *epidemic_sim*. *Epidemic_sim* écrit ce qu'il a lu dans le tube nommé qui le relie avec le programme *timer* ce qui déclenche le nouveau tour dans *citizen_manager*.

Nous avons eu un problème similaire à celui de la mémoire partagée, mais seulement sur un ordinateur (celui de Sarah), en effet si les tubes nommés n'étaient pas nettoyés entre chaque lancement de simulation, l'ouverture ne pouvait s'effectuer. De plus, nous avons quelques cafouillages au niveau des droits de lectures et écritures des tube nommés à force de copier les fonctions de lecture et d'écriture.

Affichage du graphique

Le gestionnaire d'épidémie utilise le fait que la ville est dans la mémoire partagée pour, à chaque tour, calculer différents paramètres de la ville afin de les stocker dans un fichier texte. Données qui seront utilisées par gnuplot pour générer un graph en fin de simulation.






Interface homme machine

Nous avons utilisé ncurses pour afficher les informations pendant la durée de la simulation.

Les citoyens vivent dans la ville de Disney représentée par ses cases, la couleur des cases représente le taux de contamination de celle-ci et un symbole représente le type de case.

On a alors comme légende :

Symbole	Type
H	Hôpital
F	Caserne de pompier
^	House

Couleur	Taux de contamination
	< 10%
	< 30%
	> 30%

On a aussi une représentation simple des mouvements des citoyens avec une carte similaire mais celle-ci dispose du nombre de citoyens sur chaque case. En bas à gauche on retrouve les chiffres actuels de la simulation. Et à droite la rubrique nécrologique avec le nom des personnes décédées dont le corps n'est pas encore brûlé.

Exemple d'affichage :

Attention en fonction du système d'exploitation nous avons remarqué des différences de couleurs.



On remarquera l'absence d'affichage pour l'agence de presse. En effet après moult recherches nous n'avons pas trouvé de solution pour garder un code simple et ne pas rajouter de communications inter-processus pour seulement l'affichage de celle-ci. L'utilisation de *press_agency* n'affichent donc rien malgré l'existence de communication avec *citizen_manager* via une file de messages.

La solution aurait été de créer un outil de communication entre le programme simulation épidémie et agence de presse pour récupérer l'identifiant de la

Création et lancement de programmes

Pour compiler nos fichiers nous avons décidé d'utiliser un seul makefile, nous conduisant ainsi à produire des règles générales utilisées pour produire tous les objets et les exécutables.

Le lancement de ce programme en ligne de commande peut être un peu lourd au vu du nombre d'étapes, la compilation, le nettoyage et le lancement des 4 exécutables. Nous avons alors mis un place un simple script bash qui s'occupera de toutes ces étapes pour une utilisation plus agréable de la simulation.

Intégrations extérieures

Nous avons beaucoup utilisé StackOverflow pour différents petits ou grands problèmes rencontrés. Et comme dit plus haut nous avons aussi beaucoup utilisé le cours et l'exemplier de monsieur Lebre. Nous avons utilisé le tutoriel Ncurses Howto comme initiation à ncurses, vous pouvez trouver le lien de toutes les sources dans le README, section sources.

CONCLUSION

Ce projet était intéressant à développer même si long à réaliser. Il en découle une réelle satisfaction lors de la consultation de résultats concrets sur un projet en C. Nous avons pu concrétiser l'utilisation des notions vu dans le cours de systèmes d'exploitations.

