POULLAIN Jérémy
REVEL Guillaume

Systèmes d’exploitation

RAPPORT DE PROJET

Simulation d’épidémie

2020-2021

Choix de conception et de développement :

Dans ce projet, nous avons fait des choix de conceptions liés aux consignes, ainsi que d'autres choix arbitraires.

D'abord, nous avons décidé, par rapport aux consignes, que notre projet serait composé de quatre programmes principaux, qui seraient quatre processus lancés par un programme supplémentaire qui ferait office de programme principal (programme *main*). Ainsi, il suffit d'une seule commande pour lancer l'entièreté de notre simulation.

De plus, nous avons choisi de créer beaucoup de fichiers d'entête avec de nombreuses macros pour essayer de modulariser le code et de le rendre si possible réutilisable, ou du moins lisible.

Voici un schéma des inclusions des fichiers d'entête de notre projet :

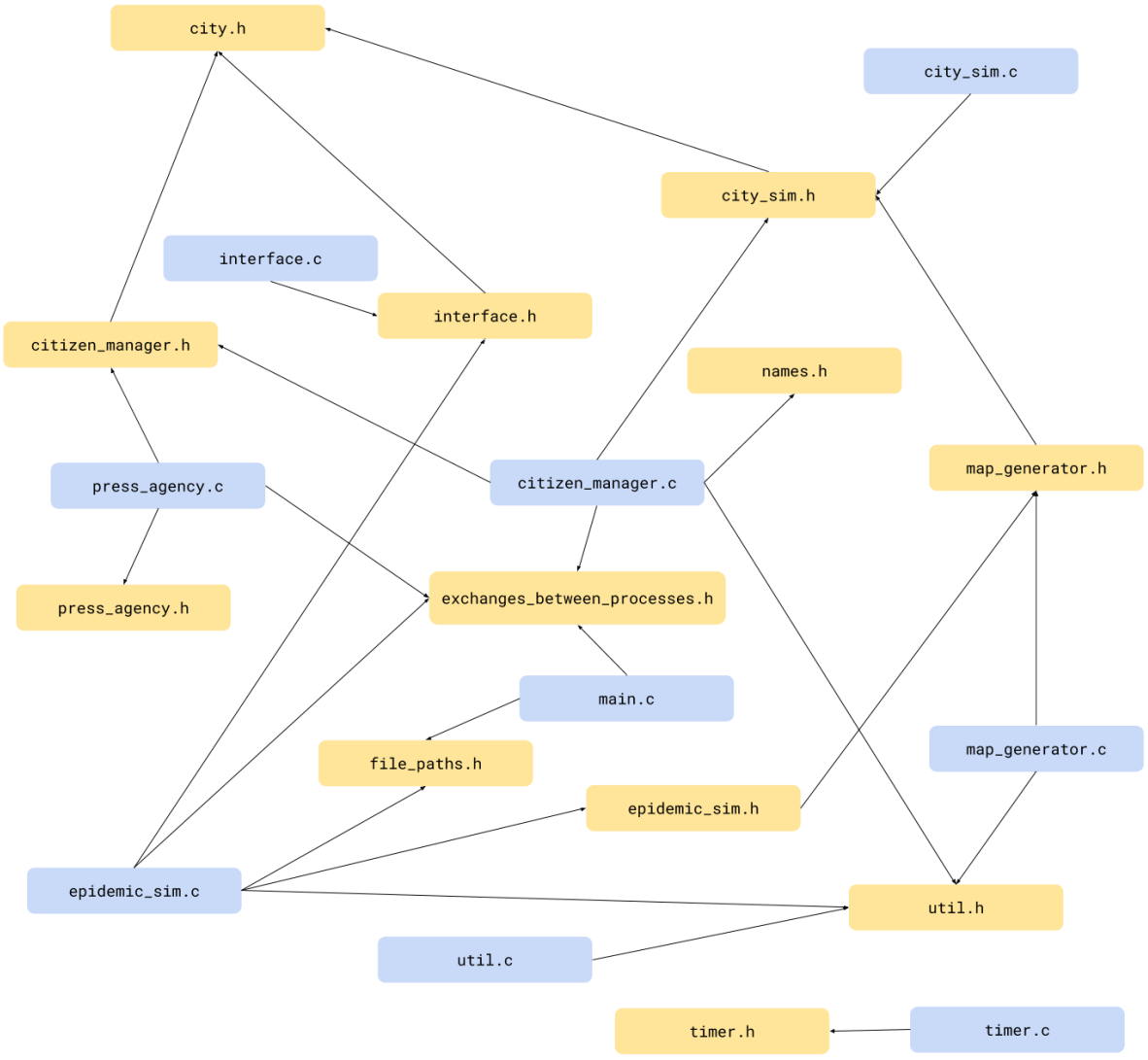
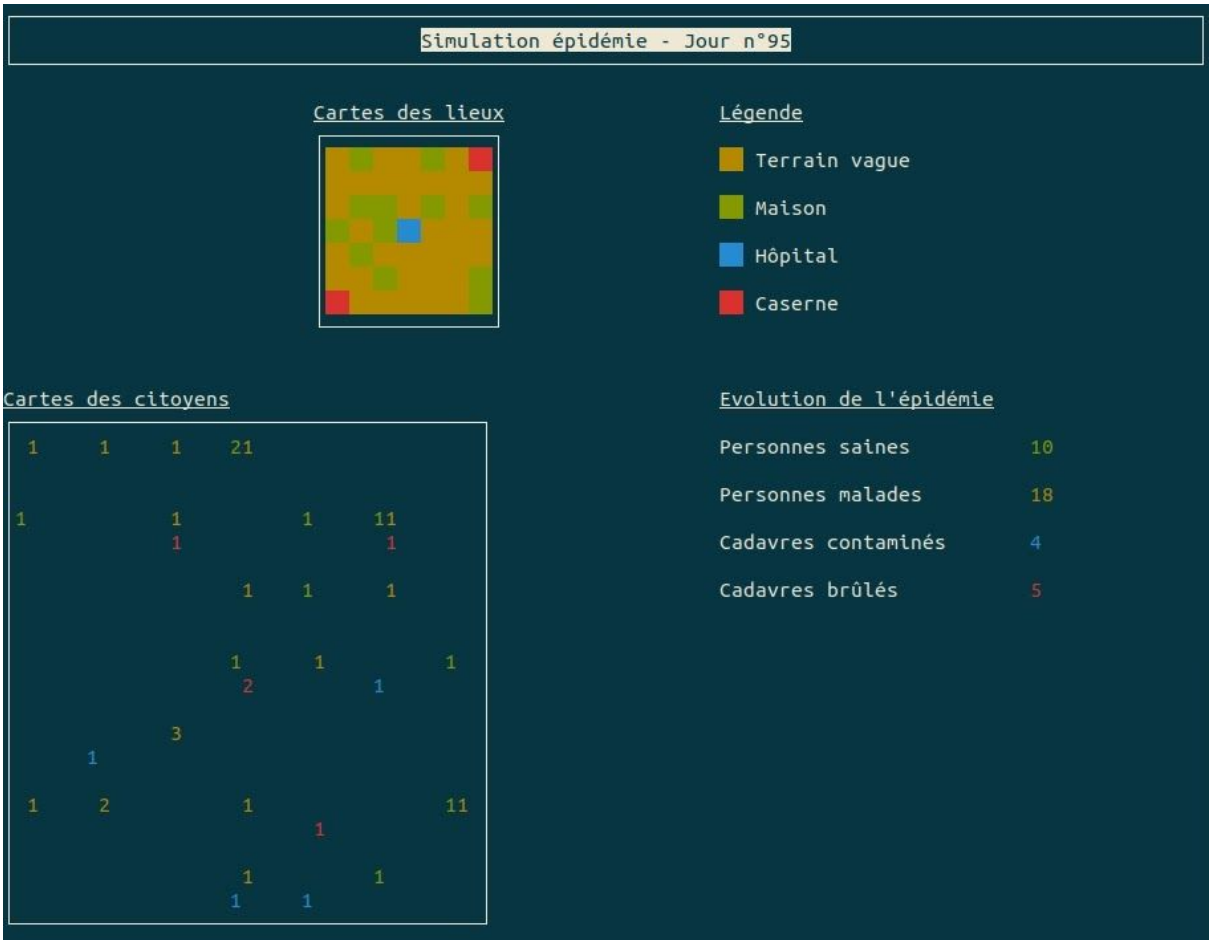


Schéma des inclusions des fichiers d'entête

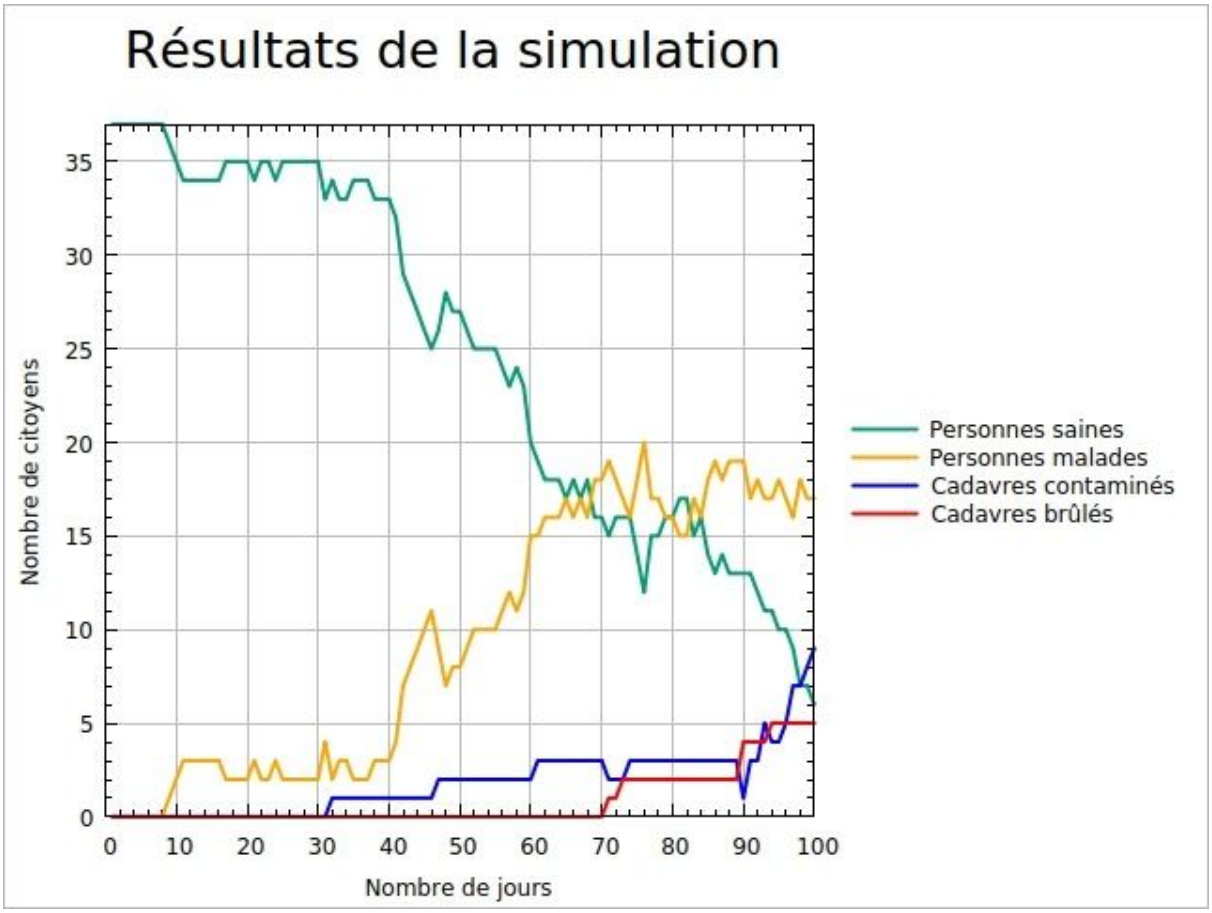
Pour représenter le déroulement de la simulation, nous avons utilisé la bibliothèque *ncurses* pour réaliser une interface qui montre à l'utilisateur différentes informations en temps réel. L'interface comporte l'affichage de la carte avec un carré de couleur correspondant à chaque type de tuile (maison, caserne, hôpital ou terrain vague).

Nous avons également choisi d'ajouter à l'interface une carte des citoyens, où l'on voit en temps réel le nombre de citoyens sur chaque case. De plus, on peut voir sur une case le nombre de citoyens dans chaque état. On peut par exemple voir sur la sixième case de la deuxième ligne qu'elle contient 1 personne en bonne santé, 1 personne malade ainsi qu'un qu'un cadavre brûlé. Un récapitulatif du nombre total de personnes dans chaque état est présent à droite de la carte.



Aperçu de l'interface en cours de simulation

En fin de simulation, un graphique résumant l'évolution de la simulation apparaît. Ce graphique, réalisé à l'aide de gnuplot, reprend les mêmes codes couleur que l'interface.



Graphique s'affichant en fin de simulation

Par ailleurs, comme ce projet était tout de même assez conséquent, nous avons dû également faire des choix quant à notre méthode de développement.

D'abord, nous avons appris les différents mécanismes des systèmes d'exploitations sous Linux, et nous avons décidé de nous limiter à cet OS pour ce projet car nous ne voulions pas augmenter la complexité de celui-ci en le rendant multiplateforme.

Nous avons codé avec la méthode agile (noms de variables explicites, fonctions intermédiaires, code explicite pour éviter la surcharge de commentaires) et nous avons aussi respecté les normes ODL et GNU comme imposé par le sujet.

En outre, afin de toujours avoir une simulation qui fonctionne et qui est testable, nous avons choisi de développer les différents modules dans un ordre précis : nous avons d'abord développé le *timer*, car cela nous semblait être le plus rapide. Ensuite, nous avons modélisé la carte, ce qui nous a servi à créer le programme de simulation *epidemic_sim*. À partir du moment où nous avons eu au moins deux processus à lancer, nous avons ajouté le programme général *main* qui lance tous les processus. Puis, nous avons ajouté les *threads* avec le programme *citizen_manager*. Après cela, nous avons intégré l'interface, qui avait été développée en parallèle des programmes précédents (nous nous étions répartis les tâches pour gagner du temps). Enfin, nous avons ajouté le graphique de fin de simulation, et le programme *press_agency* fut ajouté en dernier car il nous a semblé très indépendant de tout le reste de la simulation.

Nous avons globalement suivi les choix de conception et développement dont nous avions convenu au départ. Nous ne nous sommes pas trop éloignés de ce que nous avions prévu, cependant, nous avons rencontré plusieurs difficultés qui ont ralenti notre progression.

Difficultés rencontrées :

Dans cette partie, nous vous présenterons les différentes difficultés que nous avons rencontrées lors de ce projet, ainsi que les solutions que nous avons trouvées pour résoudre ces problèmes.

D'abord, certaines parties du projet ne nous ont pas posé de problème. Le *timer*, par exemple, n'a présenté aucune difficulté car le code déjà réalisé en cours était réutilisable quasiment dans son intégralité.

De plus, comme nous avons plusieurs processus qui s'exécutent en parallèle, il était difficile de les synchroniser pour qu'ils exécutent leurs actions dans un ordre précis. Par exemple, il ne fallait pas que *citizen_manager* appelle la mémoire partagée avant que *epidemic_sim* ne la crée. La solution a été d'utiliser les mécanismes vus en cours de systèmes d'exploitation afin de faire attendre des processus dans des boucles infinies, puis de les débloquer avec soit un signal (de *timer* vers *epidemic_sim* par exemple), soit un message via un tube nommé (de *epidemic_sim* vers *citizen_manager*).

Ensuite, nous avons choisi d'utiliser des *mutex* pour synchroniser les *threads*, mais leur utilisation nous a posé quelques problèmes au début. Nous avons réglé ces problèmes en faisant bien attention à ne pas verrouiller deux fois le même *mutex*, car cela bloque tous les *threads* qui l'utilisent indéfiniment.

Concernant l'interface, l'appropriation de *ncurses* n'a pas pris énormément de temps au début. Nous avons rapidement créé un modèle d'interface pour ensuite ajouter des fonctions qui font le lien entre celle-ci et les données de la simulation. Seulement, au moment de l'intégration, où l'on a remplacé les données par défaut par les vraies données de la simulation, nous avons eu quelques problèmes. Certaines lignes étaient décalées de façon inexplicable, cassant ainsi les fenêtres.

Résoudre ces erreurs nous a demandé beaucoup de temps supplémentaire. Nous avons d'abord trouvé les conditions d'apparition des problèmes qui semblaient d'abord aléatoires mais qui finalement contenaient une certaine logique. Par exemple, lorsqu'une ligne entière de la carte ne comportait que des terrains vagues (affichés d'une même couleur sur la carte), la ligne suivante était séparée du reste et se retrouvait tout à gauche de la fenêtre.

Pour résoudre ces problèmes, en connaissant les conditions, nous avons en quelque sorte contourné le problème en ajoutant des fonctions de *ncurses*, par exemple en ajoutant des contours à chaque fenêtre ou en rafraîchissant les fenêtres avec d'autres fonctions que celles que nous utilisions au début.

Pour une partie des problèmes que nous avons résolus, nous avons pu bénéficier de différentes aides extérieures.

Aides extérieures :

Pendant tout le processus de développement, nous avons régulièrement consulté sites internet et forums pour nous renseigner ou pour résoudre des problèmes.

Nous avons pu trouver de l'aide sur différents forums, tels que Stack Overflow (https://stackoverflow.com/) ou encore GeeksForGeeks (https://www.geeksforgeeks.org/). Nous nous sommes aussi aidés de la documentation officielle de gnuplot (http://www.gnuplot.info/) ou encore du manuel de Linux (https://linux.die.net/man/).

Nous avons régulièrement posé des questions à M. Lebret afin de mettre au clair beaucoup de points vis-à-vis du sujet. En outre, nous avons utilisé le code fourni par M. Lebret dans la section "Un point de départ sur le développement" du forum de discussion du projet pour avoir une bonne base de développement. Il était difficile de savoir par où commencer sans cette base.

Enfin, nous avons ajouté un petit easter egg à l'aide d'un ASCII-art trouvé sur le site AsciiWorld.com (http://www.asciiworld.com/-Death-Co-.html).

Conclusion :

Nous avons réussi à mener à bien ce projet en réalisant chacune des fonctionnalités demandées. Cela a pu être possible grâce à une bonne organisation, une bonne répartition des tâches, et aussi grâce à l'utilisation de Git pour mettre en commun nos codes et de Discord pour communiquer efficacement.

Ce projet fut très intéressant, il nous a appris à mettre en application beaucoup d'outils vus en cours de systèmes d'exploitation, et ce de façon ludique.