|  |
| --- |
| Rapport final |
| UE Programmation Système |
| LIAO Zuzhi - OUSSENY Irfaane - MAHAZOASY Heritiana Allan |

# 

Contenu

[I- Sauvegarde et chargement des cartes 2](#_Toc500708900)

[a) Map\_save : 2](#_Toc500708901)

[b) Map\_load : 3](#_Toc500708902)

[II- Utilitaire de manipulation de la carte 4](#_Toc500708903)

[a) Get\_width, get\_height, get\_objects et get\_info: 4](#_Toc500708904)

[b) Set\_width et set\_height : 4](#_Toc500708905)

[c) Setobject : 6](#_Toc500708906)

[d) Pruneobjects : 7](#_Toc500708907)

[III- Gestion des temporisateurs 7](#_Toc500708908)

[IV- Conclusion 9](#_Toc500708909)

# I- Sauvegarde et chargement des cartes

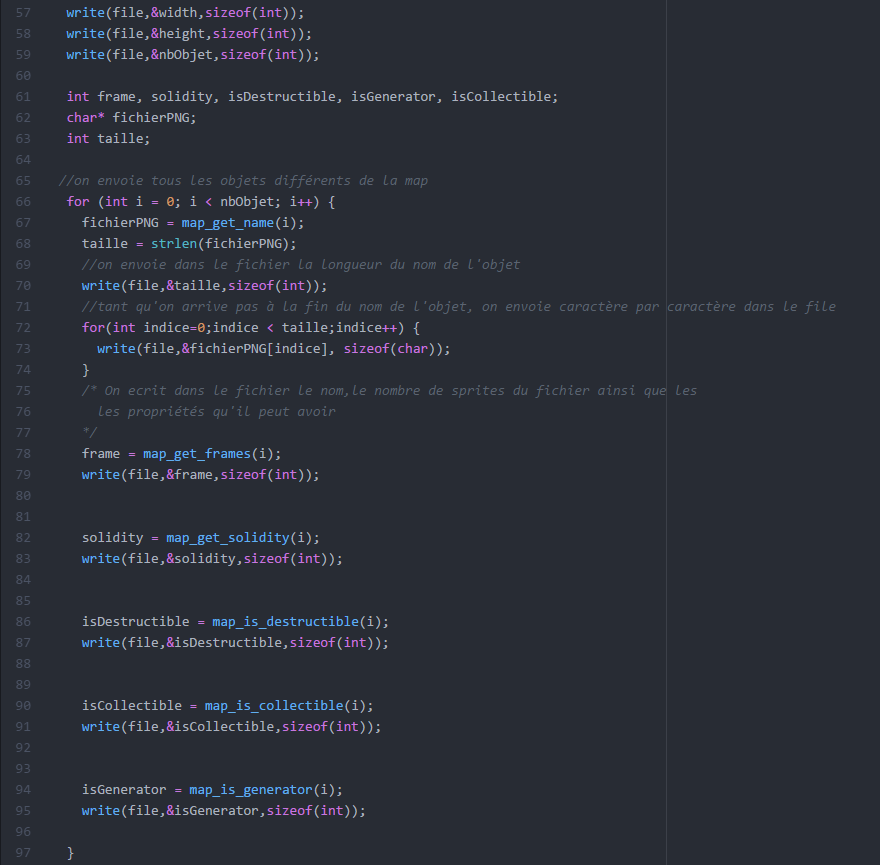
## Map\_save :

Après les premiers exercices permettant la compréhension du code de mapio.c et la gestion des objets de la map, la première tâche fut de créer les fonctions map\_save et map\_load, mettant en avant les fonctions write, read et open étudiés en cours.

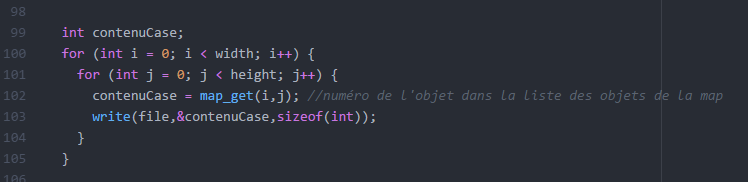
La première chose à faire a été le protocole de création d’un fichier map : l’ordre des valeurs et le contenu étant à nos soins, nous avions une grande marge de liberté vis-à-vis de la forme finale. Si au tout début cette partie a été légèrement hasardeuse, nous en sommes arrivés à ce protocole-ci :

Map\_width|Map\_height|nombre d’Objets| [taille\_nom\_objet|nom|nombre de frames|solidity|is\_destructible|iscollectible|is\_generator] \*nombre d’objets|matrice width\*height

Vous trouvez ci-dessus la structure d’un fichier map sans espaces ni les « | » et accolades qui nous permettent ici de vous donner un aperçu lisible du fichier de sortie.



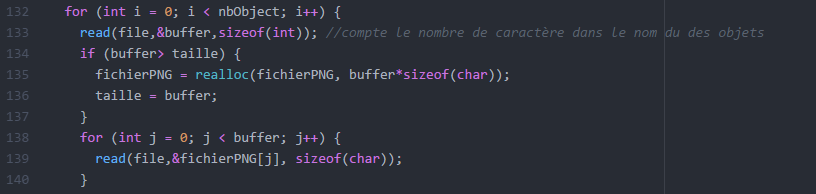
Ce n’est que plus tard que nous nous étions rendu compte du fonctionnement des objets et de la fonction map\_get. En effet, nous pensions devoir rajouter pour chaque élément de la matrice ses paramètres ainsi que son nom pour chaque case, ce qui aurait impliqué un nombre assez grand de read et de write à faire lors de la sauvegarde et du chargement. Au final, le fait de savoir qu’il suffit de renvoyer l’index de l’objet dans le tableau nous a fait beaucoup de bien.



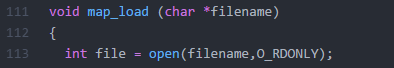
Nous avons notamment beaucoup hésité sur la forme du write et l’écriture d’un entier dans un fichier. En effet, nous ne voulions pas qu’il renvoie un entier en hexadécimal alors nous avions commencé par créer une variable string temporaire dans lequel nous écrivions l’entier via une fonction fprintf, string que nous écrivions dans le fichier afin d’être sûr que le fichier récupérerait les bonnes informations. C’est après plusieurs essaies que nous comprirent correctement l’utilisation du sizeof(int) en troisième paramètre et que cela nous aiderait d’ailleurs énormément dans la suite du devoir avec le map\_load.

## Map\_load :

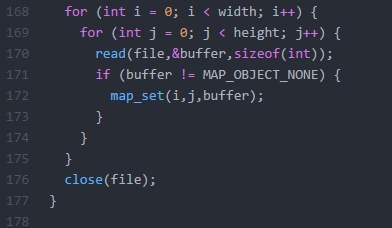
C’est en s’inspirant du Map\_new que nous avions défini le plan de base de Map\_save et de Map\_load, cette dernière n’étant que la suite logique de la fonction de sauvegarde, sa réalisation n’a pas été la plus difficile. C’est au moment où il fallait récupérer le nom des objets de la map que nous avions eu notre première réelle difficulté du devoir. En effet, à la base nous utilisions un système de flag entre les différentes informations pour être sûr que nous récupérions un mot entier. Chaque fin de nom était ponctuée d’un point d’exclamation. Si au début cela nous semblait astucieux le résultat final n’était pas celui que nous attendions et nous avions pris beaucoup de temps à le débugger sans succès. C’est à ce moment-là que l’on a décidé de restructurer map\_save afin que l’on communique directement dans le fichier la taille des mots, évitant ainsi les surprises. Les deux solutions étaient bonnes, mais l’une a été plus facile pour nous à implémenter que l’autre.



L’autre difficulté de la partie fût la bonne utilisation de la fonction open(). En effet, après avoir implémenté une fonction qui nous semblait plus que correct, ce que nous lisions dans le fichier n’était pas le résultat attendu. Pourtant tout semblait parfait du côté du map\_save. Le problème devait forcément venir de notre map\_load. A la lecture, nous recevions une erreur d’allocation à cause d’une trop grosse valeur attribuée à width et height qui étaient semblable à une adresse mémoire. Après avoir passé plusieurs demi-journées dessus, on en avait déduit que le problème devait venir de l’open qui était défini en O\_WRONLY|O\_RDONLY. C’est en enlevant la permission d’écriture et en laissant seulement la lecture que nous l’on a résolu notre problème.



Il a ensuite suffit de lire la matrice et utiliser la fonction map\_set prédéfini pour chaque point de la map pour nous retrouver avec un load fonctionnel.



# II- Utilitaire de manipulation de la carte

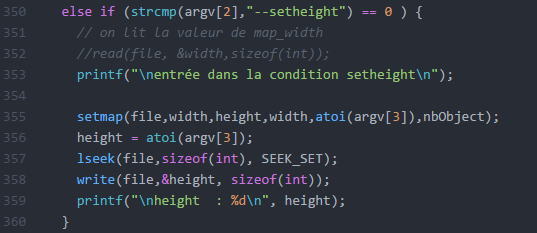
## Get\_width, get\_height, get\_objects et get\_info:

Il est clair que nous ne passerons pas beaucoup de temps dans cette section, voilà pourquoi nous regroupons les get ensemble.

En simple, nous avons d’abord récupéré toutes les informations de base, qu’importe la demande de l’utilisateur puis nous les redistribuons. La seule difficulté a été encore une fois l’appel à open que nous avions défini sur WR\_ONLY|RD\_ONLY qui donnait des valeurs hasardeuses aux reads, que nous avons ensuite transformée en O\_RDWR et qui cette fois-ci apportait les bonnes valeurs.

## Set\_width et set\_height :

On sait que le début d’un fichier de sauvegarde est composé de plusieurs entiers. Afin de remplacer une donnée, il suffit de faire un certain nombre de saut en fonction de la donnée recherchée à l’aide de lseek.



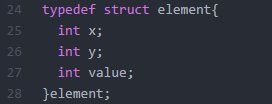
Ici, SetHeight implique de faire d’abord un saut d’entier avant d’être à la position du height et ensuite le remplacer par la valeur désirée. Il va de même pour la fonction combinant set\_width et set\_height.

Setmap est la fonction que nous avons créé afin de répondre aux besoins des différents set. En effet, outre le fait de remplacer la valeur de height ou de width, il fallait aussi réarranger la taille de la matrice du terrain. Elle prend en paramètre les nouvelles tailles et les anciennes ainsi que le fichier.

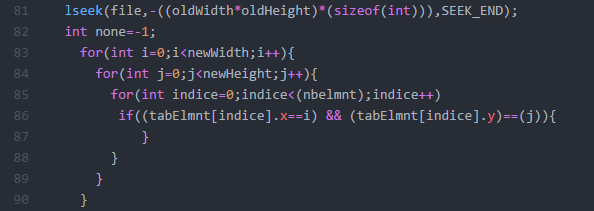
Tout d’abord, on se met au niveau du début de la matrice avec un lseek démarrant de la fin afin de revenir [largeur\*hauteur] en arrière. 

On récupère les anciennes tailles que l’on va comparer avec les nouvelles afin de savoir si l’on réduit ou agrandit la map. Grâce à ça, on a une idée de ce qui doit être fait au niveau de la réorganisation des éléments.

On a créé une structure Element qui prend des coordonnées et une valeur représentant le contenu (l’objet ) qui lui corresond et on en fait un tableau de la taille de la matrice.



On parcourt ensuite toute la matrice et on récupère chacun des élements que l’on ajoute dans le tableau.

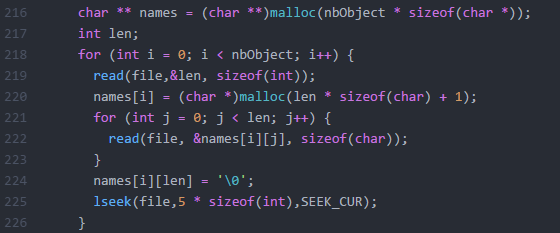


Cependant, cette fonction ne marche pas correctement pour deux raisons. La première est que les éléments perdus restent dans le fichier et l’on n’a pas réussi à les enlever, même avec un truncate. Du coup, certains éléments perdus reviennent et créent conflits avec la nouvelle map et rien ne semblait pouvoir les effacer. Le second problème vient du réajustement en hauteur qui crée une déformation du terrain malgré les réajustements réalisés.

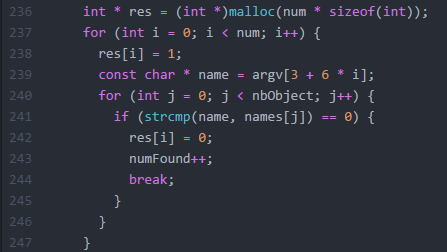
Il a été pensé qu’en passant par un fichier temporaire puis une concaténation en remplaçant toute la matrice au lieu de la réécrire par-dessus, la fonction aurait été fonctionnelle. C’est une leçon pour la prochaine fois.

## Setobject :

Tout d’abord, on crée un tableau qui prendra le nom de tous les objets de la map.



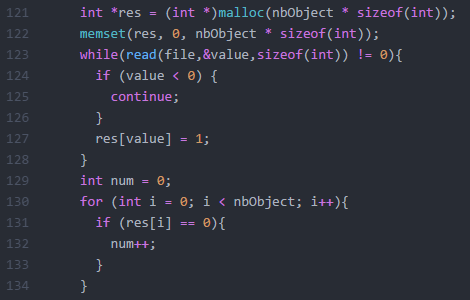
On vérifie que l’objet donné en paramètre n’existe pas déjà, auquel cas on ne fait rien.

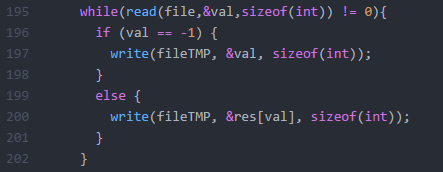


On crée un fichier temporaire dans lequel on va écrire la taille, le nom et toutes les informations de l’objet que l’on concatène ensuite au fichier principal.

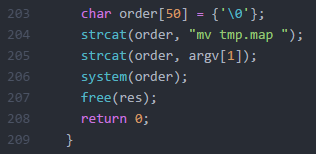
## Pruneobjects :

On crée un fichier temporaire. Une fois arrivé dans la partie map du fichier, on compte les différents éléments de map et on compte leurs récurrences.



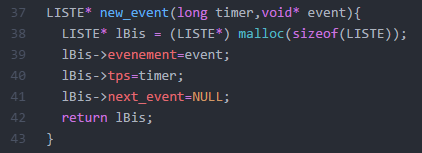
Si la récurrence de l’objet est à 0 on ne le rajoute pas dans le fichier temporaire : 

Et on concatene au fichier principal.



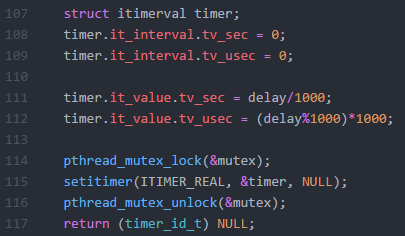
# III- Gestion des temporisateurs

La gestion des temporisateurs a été un moment assez compliquée car en dehors des parties très copier-collées tel que les initialisations des handlers et des threads, du timer et du masque, nous nous sommes retrouvé face à un bug dont nous n’avons toujours pas trouvé la source.

Nous avons créé une structure LISTE : 

C’est grâce à cette structure que nous gérons les différents événements sur la map. Nous avons ensuite implémenté une fonction new\_event qui crée un nouvel évenement. La fonction untilNext permet, elle, d’atteindre le dernier élément de la liste sans déranger à l’ordre chaîné. A l’arrivé d’un nouvel élément, si la liste d’événement est vide on ajoute un event, sinon on utilise UntilNext pour l’ajouter à la fin.

Nous en arrivons à l’initialisation du timer et l’appel à la fonction setitimer qui a été facile d’utilisation grâce aux différentes ressources sur le net.



Le lock du mutex vient de l’idée que peut-être que sur plusieurs threads, il était possible que le compte à rebours d’un timer puisse être influencé par un autre thread. Cette idée nous est venue à cause du bug auquel on a fait face. En effet, lorsque plusieurs événements s’enchaînent sur la liste chaînée alors le dernier événement ne se déclenche pas. Etonnamment, le SIGALARM qu’il devrait envoyé est interrompu ou récupéré par les autres éléments, laissant une mine ou une bombe en l’attente d’un signal qui ne vient jamais ou seulement si l’on prépare une nouvelle bombe. De ce fait, le timer\_set de notre projet est non fonctionnel malgré tout ce que l’on a essayé pour débugger ce problème.

De l’autre côté, le handler envoie sdl\_push\_event à chaque SIGALARM et fait avancer la liste chaînée.

# 

# IV- Conclusion

Le projet était une belle application de tout ce qui a été appris durant ce semestre. Pour qui est de la situation finale globale, il est clair que nous ne sommes pas fiers du résultat. Le projet a été pris au sérieux mais nous avons mal structuré notre manière de travailler, attaquant trop tardivement le projet car nous n’avions pas su balancer avec les autres matières. Par ailleurs, on dénote un certain manque de pratique de chacun dans la compréhension des différents chapitres. Néanmoins, ce projet nous a permis de mettre en avant des lacunes importantes de chacun, par rapport à la quantité de travail à fournir dans la formation.

Merci à vous pour l’encadrement et l’attention que vous avez apporté à notre travail.