TEXEREAU Gatien – FERREIRA Yohann – LAGACHE Agathe

CESI.EXIA - Première Année



Rapport de projet système et programmation procedurale

Table des matières

[I. Analyse du besoin 2](#_Toc469595031)

[1. Lanceur eXiaSaver 2](#_Toc469595032)

[2. Historique et statistiques 2](#_Toc469595033)

[3. eXiaSaver de type statique 2](#_Toc469595034)

[4. eXiaSaver de type dynamique 2](#_Toc469595035)

[5. eXiaSaver de type interactif 3](#_Toc469595036)

[6. Variables d’environnement 3](#_Toc469595037)

[II. Organisation 3](#_Toc469595038)

[III. Technique 4](#_Toc469595039)

[1. Structure de notre travail 4](#_Toc469595040)

[2. Organisation des fonctions 4](#_Toc469595041)

[IV. Bilan du projet 10](#_Toc469595042)

[1. Individuel 10](#_Toc469595043)

[2. Groupe 10](#_Toc469595044)

[3. Bilan technique 11](#_Toc469595045)

# Analyse du besoin

Pour ce projet, nous avions pour demande un exécutable lançant trois écrans de veille possédant chacun des caractéristiques particulières. Les points suivants donnent la composition et le rôle de chaque partie du livrable.

## Lanceur eXiaSaver

Le fichier principal de l’eXiaSaver est son lanceur chargé de :

* Vider la console,
* Choisir aléatoirement une des trois types d’écrans,
* Lire les variables d’environnement correspondantes au type choisi,
* Choisir un fichier image dans le répertoire des images de façon aléatoire si type d’écran est statique,
* Lancer l’exécutable correspondant au niveau choisi en passant les paramètres nécessaires (par exemple, le nom de l’image pour le type écran de veille statique, et la position initiale de l’avion pour le type interactif).

## Historique et statistiques

Un fichier sera créé et enregistrera l’historique des lancements depuis le lanceur eXiaSaver. Il devra stocker :

* La date et l’heure de l’exécution,
* Le type d‘écran lancé,
* En fonction du niveau :
  + Le nom de l’image affichée pour le niveau statique,
  + La taille de l’horloge pour le niveau dynamique,
  + La position initiale de l’avion pour le niveau interactif.

A partir de ce fichier pourront être obtenues des statistiques lisibles lors du lancement de l’eXiaSaver, grâce à un menu possédant plusieurs affichages possibles comme par exemple :

* La date,
* Le type d’écran lancé.

## eXiaSaver de type statique

L’eXiaSaver de type statique doit afficher le contenu d’un fichier PBM choisi aléatoirement au centre de la console.

L’écran de veille prend fin lors de l’appui sur n’importe quelle touche du clavier.

## eXiaSaver de type dynamique

L’eXiaSaver de type dynamique doit afficher l’heure actuelle de l’ordinateur sous le format HH:MM:SS grâce à plusieurs fichiers PBM accolés et centrés sur l’écran.

Un message doit apparaître au bas de l’écran déclarant « Cet écran sera actualisé dans quelques secondes » avec, à sa suite, un point s’affichant toutes les secondes jusqu’à actualisation de l’heure. L’intervalle de temps entre chaque actualisation doit pouvoir être paramétrée depuis un fichier.

L’écran de veille prendra fin lors de l’appui sur la barre « espace » du clavier.

## eXiaSaver de type interactif

L’eXiaSaver de type interactif doit afficher un avion apparaissant à un certain endroit de l’écran. Par défaut, l’avion apparaîtra au centre, mais l’utilisateur pourra choisir de l’afficher à une position particulière.

Cet avion se déplacera sur la console dans une direction aléatoire, mais l’utilisateur pourra changer sa direction en appuyant sur les flèches directionnelles du clavier. Quand l’avion atteindra l’un des bords de l’écran, il doit ressortir du côté opposé.

L’écran de veille prendra fin lors de l’appui sur la barre « espace » du clavier.

## Variables d’environnement

Le paramétrage de toutes les applications se fera à travers des variables d’environnement Linux saisies dans le terminal ou insérées dans le fichier «.profile » de la session utilisateur.

Ci-dessous se trouve la liste de toutes les variables d’environnement demandées :

* EXIASAVER\_HOME : répertoire où se trouvent les 3 exécutables.
* EXIASAVER1\_PBM : répertoire où se trouvent tous les fichiers PBM pour le niveau statique. Elle sera utilisée par le lanceur pour parcourir tous les fichiers du répertoire et en prendre un de façon aléatoire.
* EXIASAVER2\_PBM : répertoire où se trouvent tous les fichiers PBM pour le niveau dynamique.
* EXIASAVER2\_TAILLE : taille d’affichage de chiffres dans l’horloge numérique.
* EXIASAVER2\_SLEEP : nombre de secondes entre 2 rafraichissements de l’horloge pour le niveau dynamique
* EXIASAVER3\_PBM : répertoire où se trouvent tous les fichiers PBM pour le niveau interactif.

# Organisation

|  |  |
| --- | --- |
| Nom | Tâches |
| TEXEREAU Gatien | Programmation de l’eXiaSaver dynamique |
| Construction des structures à utiliser |
| Management de l’équipe |
| Rédaction du rapport |
| Paramétrage des bibliothèques et des fonctions correspondantes |
| FERREIRA Yohann | Création et paramétrage du lanceur |
| Gestion des variables d’environnement à l’aide d’un script |
| Création de l’écran statique |
| Création de l’écran interactif |
| LAGACHE Agathe | Création de la base de données PBM |
| Création de l’historique |
| Ecriture du rapport |

La répartition du travail n’était pas répartie de cette matière au départ du projet, mais avec certaines tâches qui se sont terminées plus vite que d’autres, nous avons décidé d’apporter du changement au planning en accord avec tous les membres du groupe afin de n’avoir personne en attente et ainsi progresser plus vite.

# Technique

## Structure de notre travail

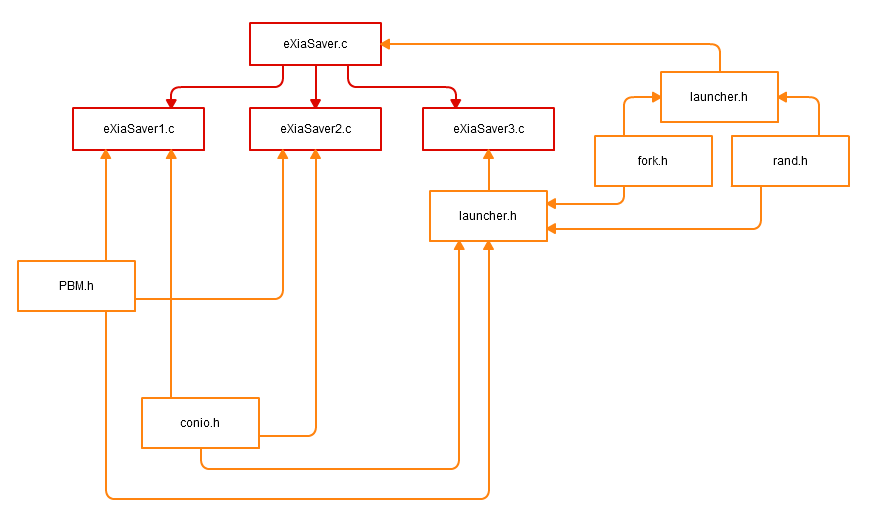
Pour notre projet, nous avons finalement utilisé 4 programmes faisant appel au total à 6 bibliothèques qui n’existaient pas avant (.h que nous avons créé) et 2 structures de données.

Nous avons principalement utilisé des tableaux dynamiques à deux dimensions et des tableaux de pointeurs. Nous avons ensuite utilisé quelques fichiers .sh qui sont des scripts qui nous ont permis par exemple de compiler tous nos exécutables en même temps ou encore de gérer les variables d’environnement.

Enfin, nous avons, comme demandé, programmé sous un environnement Linux, et nous avons principalement utilisé le logiciel *Code::Blocks*, le terminal d’Ubuntu et *Git*. Cet environnement était généré dans une machine virtuelle émulée à l’aide du logiciel *VirtualBox*.

## Organisation des fonctions

Notre projet en C possède la structure suivante :

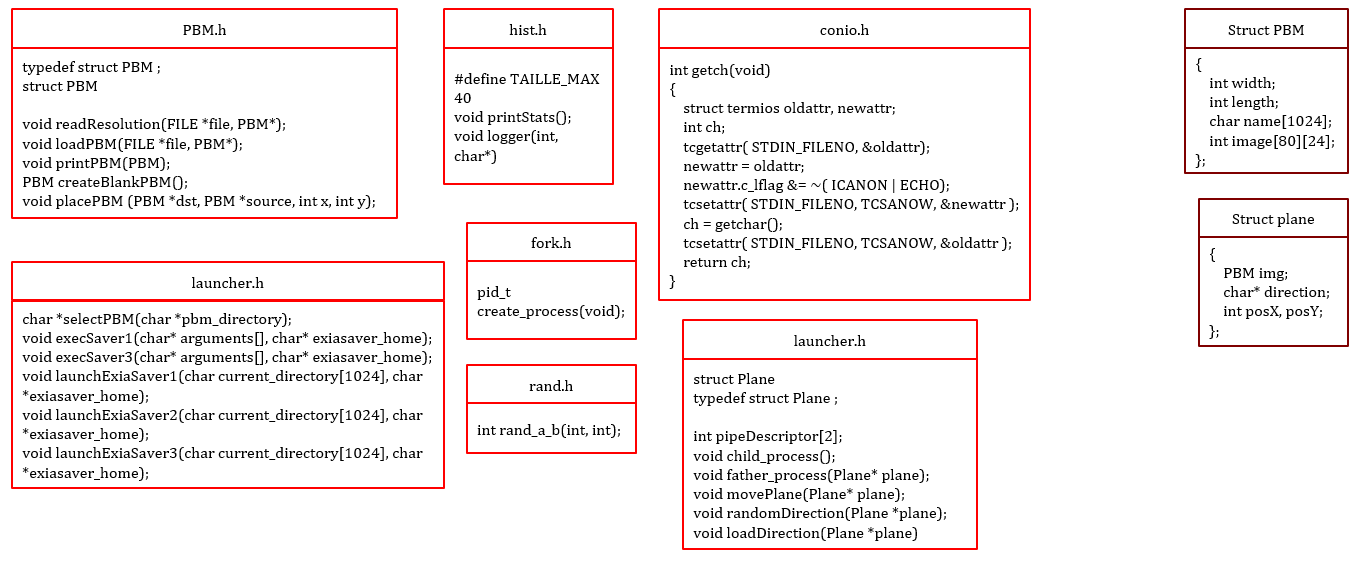


*Fig.1 : Structure du projet*

En rouge, nous pouvons voir les fichiers exécutables, qui sont composés de notre code. En jaune, nous retrouvons les différentes bibliothèques utilisées.

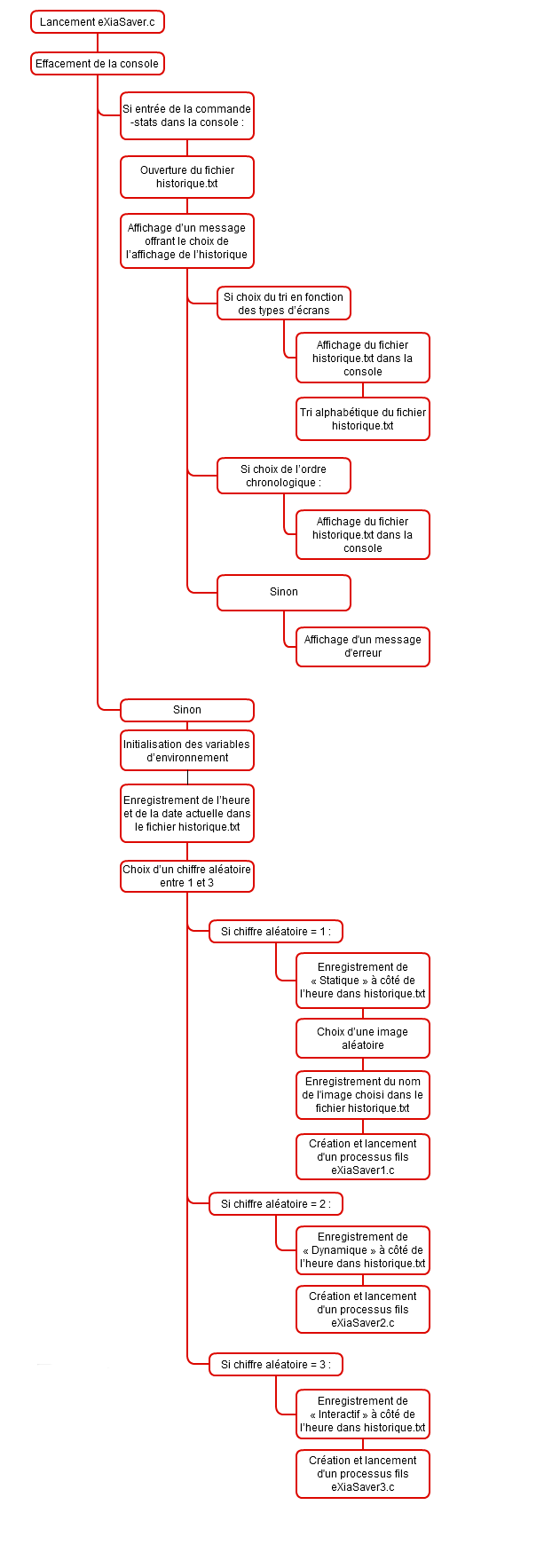
Les flèches, elles, représentent un appel de processus par un autre processus dans le cas d’une flèche rouge, ou l’utilisation d’une bibliothèque par un processus dans le cas d’une flèche jaune. Nous pouvons noter que certaines bibliothèques sont elles-mêmes composées d’une ou plusieurs autres bibliothèques.

Voici ensuite la décomposition de ce premier schéma, en plusieurs autres :

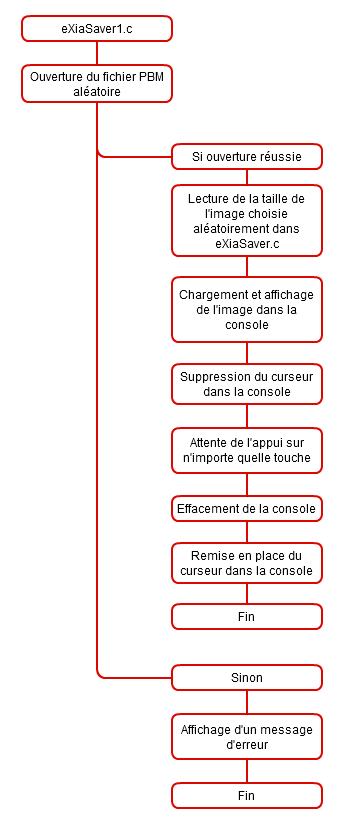


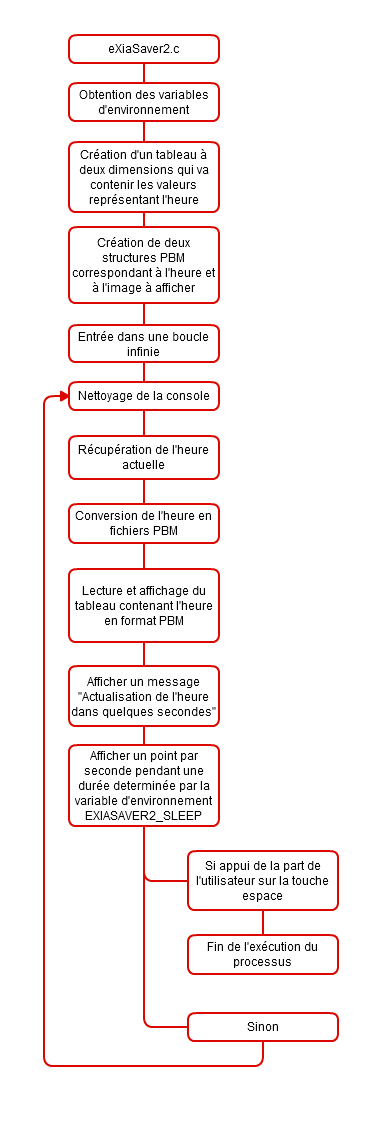
*Fig.2 : Les bibliothèques et structures utilisées pendant ce projet*

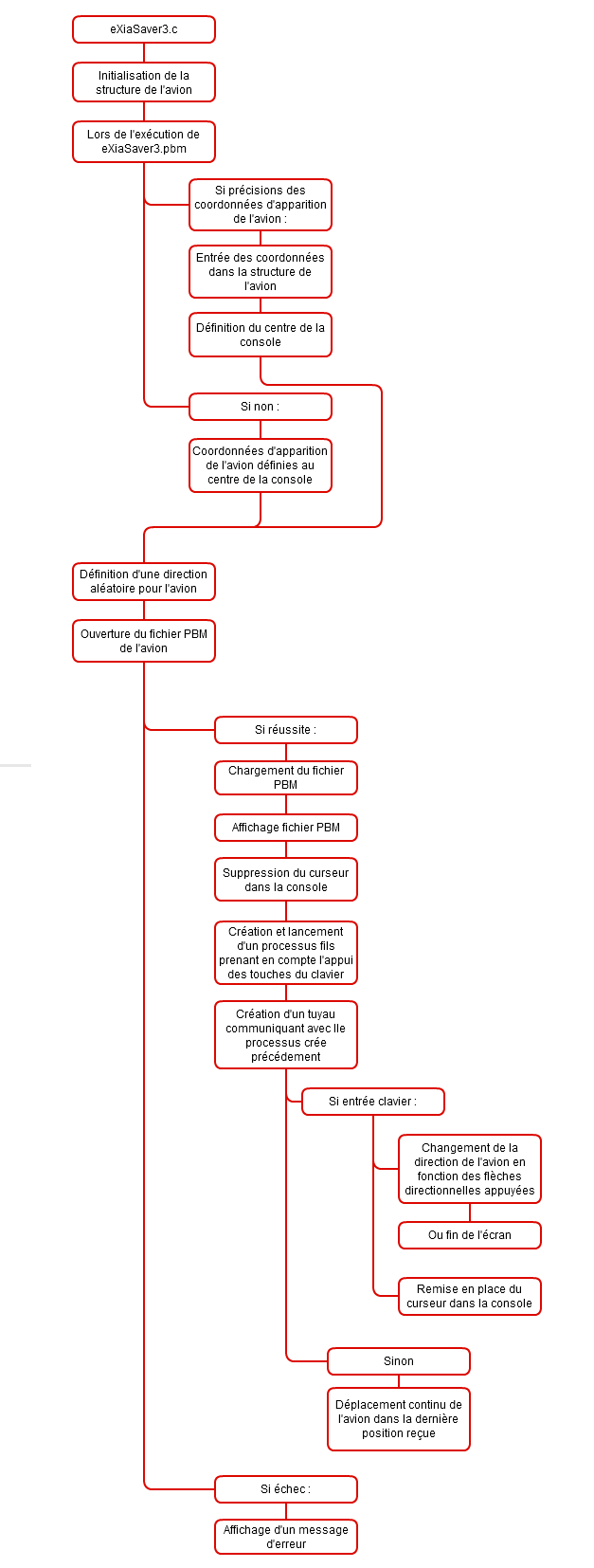
Nous voyons donc ici la composition des bibliothèques et structures utilisées pendant notre projet. Nous avons séparé bibliothèques et structures par souci de lisibilité, les structures étant des parties des bibliothèques.

*Fig.3 : Décomposition du programme du lanceur*

Nous voyons ici l’algorithme du lanceur : ce sont les actions qu’il va effectuer lors de sont exécution.

*Fig.4 : Décomposition du programme statique*

*Fig.5 : Décomposition du programme dynamique*

*Fig.6 : Décomposition du programme interactif*

# Bilan du projet

## Individuel

* TEXEREAU Gatien (Chef de projet) :
  + Équipe assez difficile à gérer car des niveaux inégaux et caractères très hétérogènes.
  + Quelques appréhensions sur la dose de travail à fournir avec le niveau d’aide disponible.
  + Quelques appréhensions au niveau de la réaction des coéquipiers qui se sont avérées vraies.
  + Pour le travail fourni personnellement, j’ai fourni une part trop peu importante du total du travail à mon goût, mais certains obstacles assez difficiles à surmonter m’ont ralenti. J’ai donné mon maximum, et ne suis donc pas déçu du résultat final de ce projet.
  + En point à améliorer, je pense que je perds beaucoup trop de temps lorsque je tombe sur un problème qui me bloque. J’ai trop tendance à chercher trop loin une solution qui est pourtant juste à portée.
  + En général, le projet était intéressant et plaisant, de plus, la découverte du travail en équipe lors de programmation en groupe et la gestion d’équipe (bien que très basique) m’ont appris beaucoup au niveau des méthodes de travail.
* FERREIRA Yohann : Au début, le projet nous a paru impressionnant et nous n'étions pas totalement capables d'identifier toutes les structures et fonctions que nous allions utiliser. Nous nous sommes fait une idée de plus en plus précise du projet au fur et à mesure de notre avancée. En ce qui concerne le groupe, l'entente était bonne et nous avons tous travaillé dur sur le projet malgré nos différences de niveau. J'ai beaucoup contribué à l'avancement du projet, mais j'aurais aimé travailler plus car je n'ai pas voulu refuser mon aide aux autres groupes, j'ai donc été énormément dérangé lors de la semaine de projet. Je pense pour ma part que les objectifs du projet ont été atteints et j'en suis plutôt satisfait.
* LAGACHE Agathe : Beaucoup d’appréhension sur le projet, j’ai eu le sentiment d’être perdue durant toute la durée de celui-ci. Je ne pense pas avoir fourni une quantité ni une qualité de travail conséquente en raison de mes nombreuses lacunes en programmation. J’aurais pu m’améliorer en demandant davantage d’aide. Notre groupe manquait de communication et d’organisation.

## Groupe

* + Quelques problèmes de cohésion au sein de l’équipe, mais qui se sont assez vite corrigés.
  + Une répartition du travail qui était censée être égale et qui s’est vite déséquilibrée suite à une vitesse d’avancement très différente en fonction des membres de l’équipe.

## 3. Bilan technique

Au final, nous avons réussi à créer notre lanceur qui va lancer aléatoirement un des trois écrans de veille qui ont eux aussi été programmés. Nous avons utilisé les variables d’environnement et respecté nos contraintes.

Nous ne retrouvons pas de problèmes d’exécutions dans notre programme et le tout est utilisable et certains paramètres peuvent même être changés par l’utilisateur (Par exemple le temps entre l’actualisation de l’heure dans l’économiseur de type dynamique en changeant la valeur de la variable d’environnement).

Nous pouvons donc considérer notre objectif atteint.