eXiaSaver

To be “*ScreenSaver*” or not to be. This is the project !

THIBAULT Benjamin
DRUON Alexandre
APPRILL Maximilien

CESI exia

Table des matières :

I] Introduction :

1) Présentation du projet :

MATRIX a lancé un appel d'offre mondial pour aider ses chercheurs à la réalisation d'un nouveau type d'écran de veille sur Linux en mode Console.

2) Analyse du besoin :

Créer un exécutable ExiaSaver qui lancera un termSaver depuis la ligne de commande d'un terminal GNU/Linux. Le termSaver peut être de trois types différents :

- Type statique
- Type dynamique
- Type interactif

Le type statique doit être capable d'afficher aléatoirement une image qui sera centrée au milieu de l'écran de veille. Le type dynamique doit être capable d'afficher l'heure actuelle au centre de l'écran, sous un format de lettre précis et la période d'actualisation de l'heure doit pouvoir être définie par l'utilisateur à l'exécution du programme. Le type interactif doit quant à lui permettre de déplacer une figure (ici, un avion) dans la console.

Le choix de l'écran affiché est effectué aléatoirement par le lanceur ExiaSaver, et il faut également mettre en place un système d'historique capable de garder en mémoire le nom du fichier exécuté, la date de lancement, le type d'écran affiché, la taille d'affichage pour le cas du type dynamique ainsi que la position initiale pour le type interactif.

3) Composition du groupe :

Le groupe est composé de :

- Alexandre DRUON
- Maximilien APPRILL
- Benjamin THIBAULT

4) Répartition des rôles :

Chef de projet : Alexandre DRUON

Il s'occupe de la répartition des tâches et doit superviser le projet sous ses différents aspects. Il doit également réaliser lui-même un nombre de tâches au sein du projet.

Programmeurs : Maximilien APPRILL et Benjamin THIBAULT :

Ils s'occupent de la réalisation du projet et de la création des différentes fonctions nécessaires aux programmes. Ils doivent, avec le chef de projet, mettre en place une réponse au besoin initiale.

II] Organisation :

1) Tâches :

Nous avons dû réaliser un ensemble de tâches tout au long du projet :

- Réaliser une feuille d'avancement détaillant l'intégralité de l'organisation de notre projet.
- Créer un écran de veille statique affichant une image centrée sur la console.
- Créer un écran de veille dynamique affichant l'heure sous un format précis toujours au centre de la console.
- Créer un écran de veille interactif permettant de déplacer un avion sur la console.
- Créer un lanceur permettant d'exécuter aléatoirement l'un des trois programmes cités précédemment
- Créer un historique qui est également lancé par le lanceur et qui stocke toutes les données relatives aux programmes précédents.
- Créer un ensemble de fichier de type PBM contenant les images à afficher dans nos programmes et fonctions.

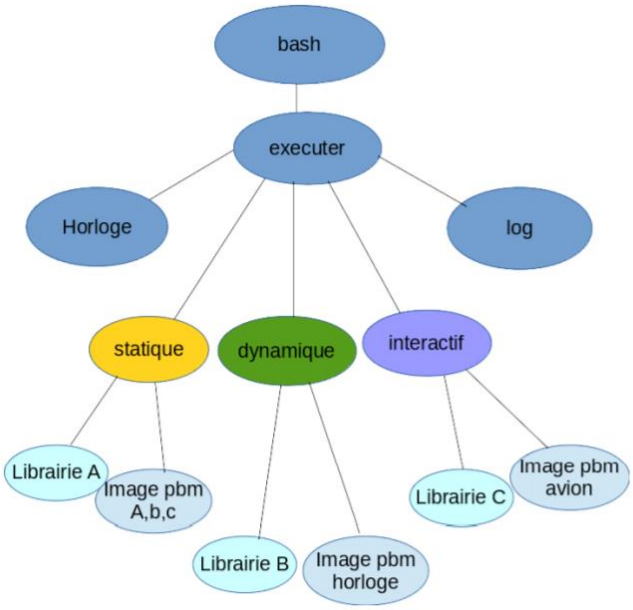
2) Planning :

Nous avions initialement prévu un planning mais celui-ci était très mal organisé et nous avons donc décidé de ne pas le suivre et de nous organiser au fur et à mesure de l'avancement du projet ce qui fut l'laborieux.

III] Choix techniques et réalisation du projet :

1) Constitution de l'architecture logiciel :

Nous avons dessiné notre conception de l'architecture logiciel que nous allions réaliser :



Notre architecture est composée de la façon suivante :

- L'exécuteur (ExiaSaver) exécute aléatoirement l'un des trois types d'écrans.
- Il stock dans l'historique (les logs) toutes les informations nécessaires que nous avons évoquées précédemment.
- Une horloge indique la date de chaque évènement de l'historique.
- Chaque type d'écran exécute différentes fonctions afin de se mettre en place et chaque écran à besoin de librairies et de différents composants (image PBM, horloge, etc..) afin de fonctionner correctement.

L'ensemble de notre programme est codé sur le système Ubuntu.

2) Présentation de l'écran statique :

Comme cité précédemment, le programme relatif à l'affichage de cet écran doit être capable de choisir aléatoirement une image et de l'afficher au centre de la console pendant une durée indéterminée. L'arrêt de ce programme doit provoquer la disparition de l'image à l'écran

Notre programme est constitué de plusieurs fonctions :

- Une fonction capable de lire le contenu d'un fichier de type PBM
- Une fonction capable d'afficher le contenu d'un fichier PBM à l'écran
- Une fonction permettant de générer un nombre aléatoire pour ensuite afficher une image en fonction du numéro obtenue

- Une fonction capable de centrer l'image sur l'écran.
- Une fonction qui récupère l'heure système et qui l'affiche en caractères ASCII avec un taux de rafraichissement adaptable.

a) Réalisation des images :

Nous avons créé les images que nous avons utilisés dans le cas de ce type d'écran. Les images sont stockées dans des fichiers de types PBM (portable pixmap). Il s'agit d'un format graphique qui permet de convertir différents types de fichiers entre plusieurs plateformes. Le principe de la création de ce type d'image est simple : le fichier de type PBM contient le format de l'image que l'on souhaite afficher, ainsi que le « dessin » de l'image sous forme binaire, c'est-à-dire une succession de 0 et de 1 qui détermine les caractères que l'on souhaite afficher à l'écran. Le chiffre 0 représente une zone vide et le chiffre 1 représente l'affichage d'un caractère. Notre programme doit donc être capable de lire ce type de fichier et d'afficher son contenu.

Exemple de fichier PBM :

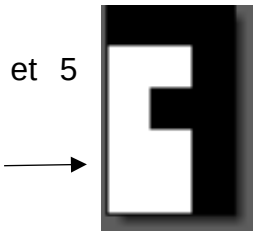
Il s'agit d'un fichier PBM représentant le chiffre « 7 ».

- P1 indique que le fichier est écrit en binaire.
- 3 et 5 sont les dimensions de l'image : 3 colonnes et 5 lignes

En lisant ce type de fichier avec une plateforme comme photo filtre ou GIMP, on obtient l'image suivante :

```
P1
3 5

1 1 1
0 0 1
0 1 1
0 0 1
0 0 1
```



b) Choix de l'image :

Le choix de l'image affichée est effectué de façon aléatoire grâce à une fonction rand() que nous avons créé. Nous avons configuré la fonction de façon à ce qu'elle affiche un nombre aléatoire comprit entre 0 et 2. Nous avons attribué un nombre pour chaque image et c'est ainsi que s'effectue le choix de celle-ci : en fonction du nombre généré par notre fonction, une image est choisie par le programme et affichée par celui-ci.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h> //Ne pas oublier d'inclure le fichier time.h

int main()
{int i, n;
    time_t t;

    srand((unsigned) time(&t));

    int r = rand() % 3;

    printf("Le nombre aleatoire compris entre 0 et 2 est: ");
    printf("%d\n", r);
}
```

```
Le nombre aleatoire compris entre 0 et 2 est: 1
```

c) Affichage de l'image :

Pour afficher l'image dans la console, notre programme doit être capable de lire un fichier de type PBM et d'afficher son contenu. Ceci est rendu possible grâce à la fonction fopen() : le programme est capable d'ouvrir le fichier PBM et de l'afficher.

Les deux premières lignes du fichier PBM (P1 ainsi que les dimensions de l'image) ne doivent pas apparaître à l'écran, nous avons donc dû filtrer ces lignes de façon à ignorer P1 et à récupérer les dimensions de l'image que nous avons stockées dans un tableau afin de les utiliser plus tard pour le centrage de l'image.

d) Centrage de l'image :

Une fois que nous avons affiché notre image dans la console, il nous reste ensuite à la centrer pour répondre à la première demande. Pour ce faire, nous avons réalisé une fonction de centrage qui prend pour paramètres des coordonnées x et y.

Cette fonction permet de récupérer la taille de la console au démarrage d'un programme (le nombre de lignes et le nombre de colonnes composant la console) puis de centrer cette image en fonction des informations récupérées par la fonction lors de l'exécution du programme. Ceci permet non seulement de centrer l'image dès l'exécution du programme mais également cas de modification de la taille de la console pendant l'exécution du programme : en effet, il est possible d'agrandir la console pendant l'affichage de l'image, et celle-ci se repositionnera au centre automatiquement.

Enfin, si l'utilisateur appuie sur une touche du clavier, le programme le détecte automatiquement est agi en conséquence : la console est effacée et l'image disparaît de l'écran.

3) Présentation de l'écran dynamique :

Nous devons réaliser un programme capable d'afficher l'heure locale en grand caractère au centre de la console. De plus, il faut également que le programme affiche une phrase « l'heure s'actualisera dans …. (n secondes) » et représenter chaque seconde qui précède l'actualisation par l'affichage d'un point.

Récupération, affichage et actualisation de l'heure :

Nous avons créé une fonction permettant de récupérer l'heure actuelle de l'ordinateur et de l'afficher en temps réel dans la console.

Ensuite, pour afficher celle-ci, nous avons dû créer une fonction permettant d'afficher l'heure sous forme d'image PBM. En effet, la contrainte imposée est d'afficher l'heure sous un format par défaut étant de 5x3 pixel. Or il est impossible de modifier directement la taille des caractères dans la console, c'est pourquoi notre capable doit est capable de récupérer l'heure et de l'afficher en « grand » en utilisant des fichiers de type PBM. Chaque chiffre possède son équivalent sous forme PBM, ce qui permet d'implémenter l'heure sous format image.

La fonction de centrage de l'heure est la même que celle utilisée dans le cas de l'écran statique.

Concernant l'actualisation de l'heure, nous avons créé une fonction prenant pour paramètre la fréquence de rafraichissement définie par l'utilisateur au démarrage du programme. Grâce à cet fréquence de rafraichissement, l'heure s'actualise toute les n secondes (n dépend de la fréquence).

4) Présentation de l'écran interactif :

Pour des raisons de temps et de complication, nous avons décidé de nous focaliser davantage sur l'écran dynamique et sur l'écran statique et également de ne pas traiter le cas de l'écran interactif.

5) L'historique (logs) :

L'historique des programmes se trouve dans le fichier historique. C'est un fichier qui contient plusieurs informations sur les programmes, à savoir :

- La date et l'heure d'exécution d'un programme
- Le type de programme exécuté (statique ou dynamique)
- Le nom du fichier image affiché par le programme statique
- La taille de l'horloge pour le type dynamique
- (La position initiale de l'avion pour le type interactif)

6) Le lanceur ExiaSaver :

Le lanceur ExiaSaver est un programme qui exécute aléatoirement l'un des trois types d'écran de veille. Dans notre cas, il exécuté aléatoirement le type statique ou le type dynamique.

IV] Conclusion :

1) Synthèse des problèmes rencontrés :

Nous avons rencontré différents problèmes durant le projet :

- La fonction de centrage, que nous avons dû adapter afin de center l'image en fonction de la taille de la console et non pour une taille définie.
- L'affichage de l'heure sous format PBM : en effet, il est impossible de modifier la taille des caractères de la console, il nous a donc fallu créer des fichiers PBM et les afficher en fonction de l'heure, ce qui nous a pris beaucoup de temps.
- L'impossibilité d'utiliser certaines fonctions de Windows sous Linux : nous avons dû nous adapter à cette contrainte qui a été présente tout au long du projet.
- Une mauvaise organisation et répartition des tâches qui nous à freiner en début de projet.

2) Bilan :

Tout le groupe est d'accord pour dire que bien que le projet ne fût pas facile, il était en revanche très intéressant et épanouissant. Ce projet nous a permis de découvrir le langage C sous les contraintes de Linux, ce qui nous a permis de découvrir ce langage sous un aspect différent. L'ensemble du groupe aurait cependant souhaité obtenir un peu plus de support tout au long du projet. Enfin, le fait que le codage sous Linux ne soit pas exactement le même que sous Windows nous a retardés dans la réalisation du projet, mais nous sommes conscients que nous aurions dû commencer la programmation sous Linux dès le démarrage du projet.

3) Perspective d'évolution :

Une meilleure répartition du travail et des tâches aurait été bénéfique dans l'optique de ce projet. Nous aurions également dû travailler davantage à l'aide de GitHub. Si nous avions eu un plus de temps ou si nous en avions également moins perdu nous aurions surement été capable de réaliser l'écran de veille de type interactif car nous avions réussi à obtenir les connaissances nécessaires.