Projet Sécurité
Virus Compagnon

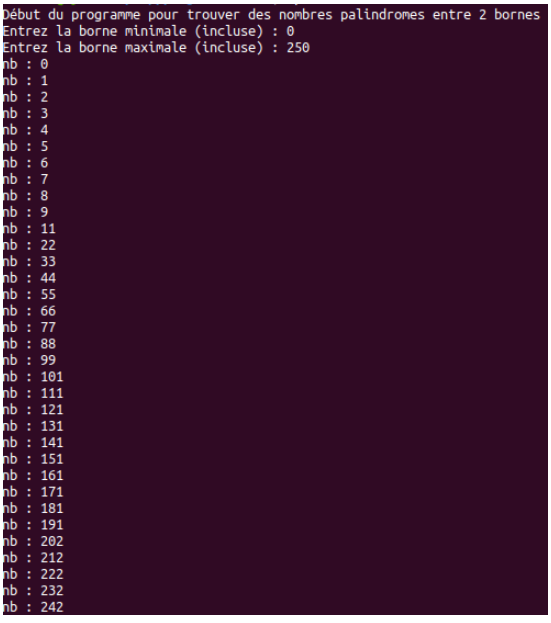
Groupe : Ilker SOYTURK – Dorian MAIDON

Table des matières

1 - Programme utilitaires

Programme 1

Le but de notre premier programme est de trouver les nombres palindromes entre deux bornes. Pour cela, nous utilisons le mode console pour interagir avec l'utilisateur. Nous demandons à l'utilisateur les 2 entiers (les bornes) qui l'intéresse, puis nous appliquons dans notre code une fonction qui s'intitule find_palindromic_number. Dans cette fonction, nous testons chaque entier compris entre les 2 intervalles (inclus) ; l'algorithme est le suivant : il suffit dans un premier temps de convertir notre entier en une chaîne de caractère, ensuite, nous retournons cette chaîne de caractère avec une fonction nommée strrevv puis on effectue une comparaison entre l'ancienne chaîne et la nouvelle. Si la chaîne est identique, on l'affiche directement sur la console (nous ne stockons pas les nombres). Voici un exemple d'exécution du programme 1 :



Programme 2

Le programme 2 permet de calculer la distance entre deux points en 2D. Nous avons juste implémenté deux types de distance : euclidienne et manhattan. L'utilisateur choisi le type de distance qu'il souhaite en ligne de commande lors de l'exécution. Par exemple :

```
dmaidon@syrie18:/tmp/projetSecurite$ ./MonPG2 euclidienne
Début du programme pour calculer la distance entre deux points
Entrez les coordonées du points A au format suivant : x1 y1
```

Ensuite, on lui demande d'entrer les coordonnées du point A (comme on peut le voir juste au-dessus) et du point B. Le programme récupère ces informations et utilise une fonction nommée dist qui réalisera les calculs nécessaires selon le type de distance choisi auparavant. On utilise dans ce programme la librairie math pour effectuer des calculs tels que les puissances ou encore les valeurs absolues..

```
dmaidon@syrie18:/tmp/projetSecurite$ ./MonPG2 euclidienne
Début du programme pour calculer la distance entre deux points
Entrez les coordonées du points A au format suivant : x1 y1
20 50
Entrez les coordonées du points B au format suivant : x2 y2
40 38
Distance : 23.323807
```

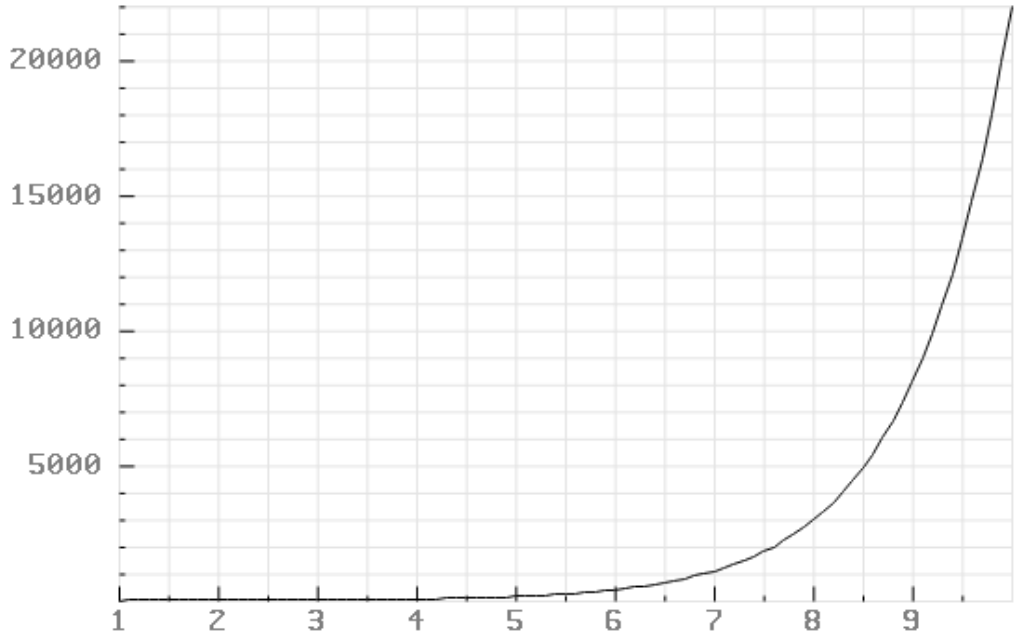
Programme 3

Dans ce programme, nous souhaitons afficher les courbes des fonctions que l'on pourrait qualifier d'usuelle en mathématique : log..fonction polynomiale de degré 2..exp etc. L'utilisateur choisit parmi 6 fonctions au lancement du programme puis, à nouveau, il rentre l'intervalle dans lequel il souhaite visualiser la courbe. La sortie de ce programme résulte en la création d'une image (sous format png). Pour cela, on utilise une fonction initialize_array qui stockera dans sa première ligne les valeurs de x puis les valeurs de y dans la deuxième. On retourne donc un tableau en 2D que l'on passe en paramètre à une fonction drawing. Cette fonction se sert d'une librairie permettant à partir de deux tableaux 1D de générer des images montrant des courbes de fonction.

Voici un exemple :

```
dmaidon@syrie18:/tmp/projetSecurite$ ./MonPG3
Debut du programme pour afficher les courbes des fonctions basiques en mathematiques
Entrez un nombre correspondant à la courbe que vous souhaitez voir
1-Fonction affine du type : ax + b
2-Fonction du second degré du type : ax² + bx + c
3-Fonction cosinus du type : cos(x)
4-Fonction sinus du type : sin(x)
5-Fonction logarithmique du type : log(x)
6-Fonction exponentielle du type : e(x)
6
Entrez la borne minimale pour visualiser la courbe : 1
Entrez la borne maximale pour visualiser la courbe : 10
Entrez le nom du fichier image sortant : img1
Sauvegarde faite
```

Ensuite, le fichier img1.png est disponible dans le dossier où nous avons exécuté le programme, si on l'ouvre, nous obtenons cela :



Nous souhaitions combiner cette librairie avec des expressions mathématiques plus complexes comme par exemple :

$$5 \cdot \cos\left(\frac{\pi}{2}\right) + 8 \cdot \sin\left(\frac{\pi}{4}\right)$$

Mais nous avons rapidement changé d'idée et sommes revenus sur quelque chose de plus simple (ce que nous avons actuellement). En effet, si l'utilisateur entrait une expression mathématique, l'évaluer aurait été compliqué en C.. En python, il existe des fonctions implémentées de base comme la fonction eval qui permet de transformer un string en une opération.. mais en C, il est nécessaire d'utiliser des algorithmes et nous ne voulions pas passer trop de temps à les comprendre et à les utiliser (puisque nous préférions faire quelque chose de correct/minimal dans les temps).

Programme 4

L'objectif du programme numéro 4 est de calculer les impôts (de manière simplifiée) sur les revenus . Pour cela, nous prenons en compte différents critères tels que le nombre de parts du foyer, son revenu **net** global annuel. Nous considérons que le nombre de parts est plafonné à 15. Le montant net imposable est égal au revenu net global après abattement de 10 %. Le quotient familial est ensuite calculé.

Un exemple : Si un utilisateur entre un revenu net global de 255 000 euros, avec un nombre de part* de 3, en se référant au tableau des % impositions*, nous pouvons calculer le montant de l'impôt du comme cela :

Tranche 1 : de 0 à 10 064 soit 10 064 * 0 = **0€**

Tranche 2 : de 10 064 à 25 659 soit (25 659 – 10 064) * 0.11 = **1 715.45€**

Tranche 3 : de 25 659 à 73 369 soit (73 369 – 25 659) * 0.3 = **14 313 €**

Tranche 4 : de 73 369 à 76 500 soit (76 500 – 73 369) * 0.41 = **1283,71€**

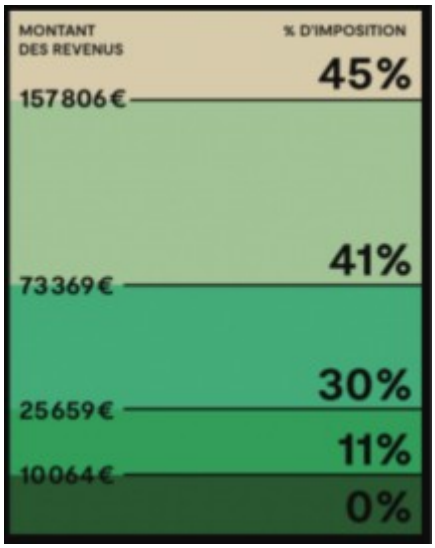
Le montant de l'impôt est ensuite obtenue en multipliant la cotisation par le nombre de parts..

17 312, 16 * 3 = 51 936,48€

*

	Nombre de parts	
Nombre d'enfants	Célibataire, divorcé ou veuf	Couple marié ou pacsé
0	1	2
1	1,5	2,5
2	2	3
3	3	4
4	4	5
par enfant supplémentaire	1	1

*



En exécutant, on obtient le même résultat :

```
dorian@dorian-virtual-machine:~/Documents/projetSecurite$ ./MonPG4
Bienvenue sur le module de calcul de vos impots !
Êtes-vous marié ?["O" pour oui/"N" pour non]
O
Combien d'enfants avez-vous ?(Entrez une valeur entière)
2
Quel est votre revenu net global ?
255000
Vos impôts s'élèvent à hauteur de : 51936.48 €
```

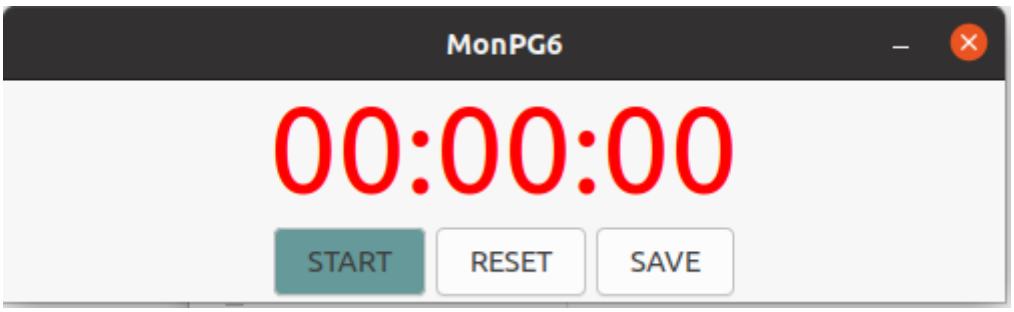
Programme 5

Dans le programme 5, nous calculons le PPCM (plus petit commun multiple) entre deux entiers qui sont passés en argument lors de l'exécution.

```
dorian@dorian-virtual-machine:~/Documents/projetSecurite$ ./MonPG5 6 13
a : 12, b : 13
a : 18, b : 13
a : 18, b : 26
a : 24, b : 26
a : 30, b : 26
a : 30, b : 39
a : 36, b : 39
a : 42, b : 39
a : 42, b : 52
a : 48, b : 52
a : 54, b : 52
a : 54, b : 65
a : 60, b : 65
a : 66, b : 65
a : 66, b : 78
a : 72, b : 78
a : 78, b : 78
Le plus grand multiple commun de 6 et 13 est 78.
```

Programme 6

Dans le programme 6, nous avons réalisé un chronomètre avec la librairie GTK pour changer du mode console. Il y a 3 boutons, un bouton permettant de lancer le chronomètre, un deuxième pour le réinitialiser (et donc arrêter le timer) et un dernier pour sauvegarder un temps si l'utilisateur le souhaite.



Chaque bouton est utilisé pour appelé une fonction (fonctionnalité) du programme. Une fonction est donc lié à un signal, la connexion se fait grâce à la fonction g_signal_connect. Aussi, si l'utilisateur souhaite mettre en pause (sans réinitialiser donc), il peut simplement appuyer sur START à nouveau.

Lorsque l'utilisateur appuie sur le bouton SAVE, nous ajoutons (ou nous créons) au fichier checkpoint.txt des données sur les temps d'enregistrements.

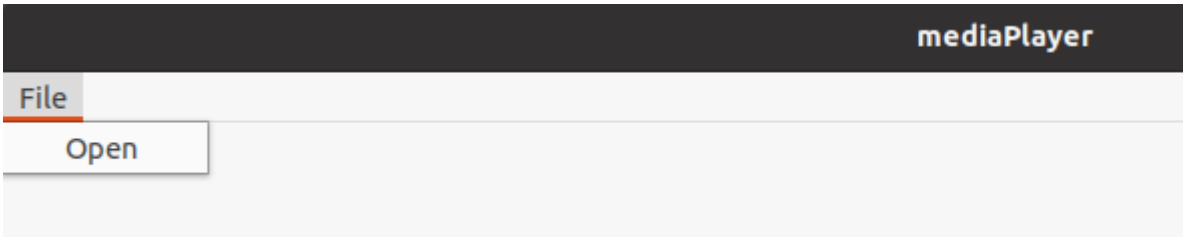
```
------------------------------
Checkpoint numero 1 : 00:01:04
------------------------------
Checkpoint numero 1 : 00:00:35
------------------------------
Checkpoint numero 1 : 00:00:02
Checkpoint numero 2 : 00:00:05
Checkpoint numero 3 : 00:00:07
```

Chaque « ----...-- » correspond à une nouvelle exécution du programme.. ainsi, c'est mieux séparé et plus clair.

2 - Mediaplayer

Partie utilitaire

Dans cette première partie du Mediaplayer (c'est-à-dire sans la partie virale) nous utilisons à nouveau GTK pour afficher des images. Dans la fenêtre graphique, nous créons un menu pour ouvrir une image/un dossier :



Ensuite, nous utilisons la fonction gtk_file_chooser_dialog_new pour que la boite de dialogue s'ouvre. Nous passons en paramètre de cette fonction le fait que nous voulons ouvrir directement un dossier (un dossier image qui pourrait susciter l'attention de la victime !) plutôt qu'une image. Ainsi, l'intégralité des images peuvent être visionnés en cliquant sur le bouton suivant.

Si une image est trop grande par rapport à la fenêtre, on les redimensionne (la fenêtre **ET** l'image) pour essayer d'avoir quelque chose de plus ou moins « propre » (et qui ne dépasse pas de l'écran si l'image est trop grande par rapport à notre matériel).. Nous avons aussi rajouté le fait que l'utilisateur puisse naviguer entre les images grâce aux flèches directionnelles de son clavier.

Processus d'infection

La situation initiale est que notre programme contagieux, est dans le même répertoire que les programmes annexes de l'utilisateur. Dans notre MediaPlayer nous avons la possibilité de choisir le nombre d'infections par exécutions :

```
//number of programs that we want to infect at once
int numberOfFilesToInfect = 2;
//calling the main function for the infecting process
renameFiles(numberOfFilesToInfect, argv[0]);
```

Après avoir choisi le nombre d'exécutables que nous souhaitons infecter lors de l'exécution de nos programmes contagieux, nous appelons la fonctions renameFiles, qui va effectuer tout le processus d'infection. Dans cette fonction nous commençons par itérer sur chacun des fichiers et dossiers présent dans le répertoire courant. Lorsqu'un fichier est de type régulier et exécutable et n'est pas le MediaPlayer, nous prenons le nom de ce fichier auquel nous ajoutons l'extension ".old", afin

d’essayer d'ouvrir un tel fichier si il existe. Si ce n'est pas le cas, nous renommons ce fichier dit exécutable en y ajoutant l’extension ".old".

Après le renommage du « vrai » programme (celui qui est inoffensif), la fonction copyFile est appelé. Cette fonction permet la copie du MediaPlayer dans un fichier portant le nom original de l'exécutable renommé. Pour cela, on crée deux fichiers, le premier pour la source et le deuxième pour la destination. Le fichier source est lu à l'aide de la fonction fread ; tant que la lecture n'est pas égale à 0, on écrit dans le fichier destination (portant le nom original du fichier cible) à l'aide de la fonction fwrite.

Finalement, après la fonction copyFile, le processus d'infection est terminé. Si nous sommes dans le MediaPlayer, l'interface du MediaPlayer sera affichée. Mais si nous lançons l'exécution d'un programme infecté, cela nous redirigera dans l'exécution du programme original avec l'extension ".old" à l'aide de la fonction system.

3 - Réponse aux questions

- *Question 1* : « Pourquoi d’après vous, dans le cas du virus compagnon cette vérification doit-être double ? »

  Nous pensons qu’il est nécessaire de faire une double vérification car tester simplement l’extension du fichier cible courant n’est pas **suffisant**. En effet, si l’extension du fichier est .exe (ou rien sous unix), il y a 2 cas possibles : le fichier est infecté OU le fichier est « sain /inoffensif». Mais pour savoir si le fichier est infecté (cas 1), nous devons donc rechercher si dans le répertoire courant il y a un fichier avec l’extension « .old »

- *Question 2* : « D’après vous, que se produit-il si l’étape 3 est oubliée ou ne fonctionne pas ? »

  Si l’étape 3 ne fonctionne pas, on ne pourra pas exécuter la partie « saine » du programme infecté car notre fichier corrompu n’est pas un exécutable, ainsi, on ne pourra pas rendre la main au vrai programme, et cela est mauvais car peut éveiller des soupçons.

- *Question 3* : « D’après vous, comment à travers cette étape (6), peut-on amplifier l’infection ? »

  D’après nous, nous pouvons amplifier l’infection lors de l’étape (6) en utilisant une boucle qui fera autant d’itération que l’on pourra définir nous-même dans le code du programme lors de l’infection.

- *Question 7* : « D’après vous, que se produit-il si l’étape 7 est oublié ou ne fonctionne pas ? »

  Si l’étape 7 n’est pas fonctionnel ou oublié, le programme « sain » ne va pas s’exécuter et cela rejoint la réponse de la question 2, c’est-à-dire : éveiller les soupçons de l’utilisateur.

4 - Conclusion

Pour conclure, le projet nous a permis de retravailler sur des éléments « basiques » du C dans un premier temps, cela a été grandement utile. Aussi, nous avons pu explorer des outils pour les interfaces graphiques telles que GTK (dans notre cas), domaine en C dans lequel nous étions tous les deux néophytes. D'un point de vue virologie du projet, travailler sur le développement d'un virus compagnon nous a permis de mieux comprendre son fonctionnement et son « déploiement ».