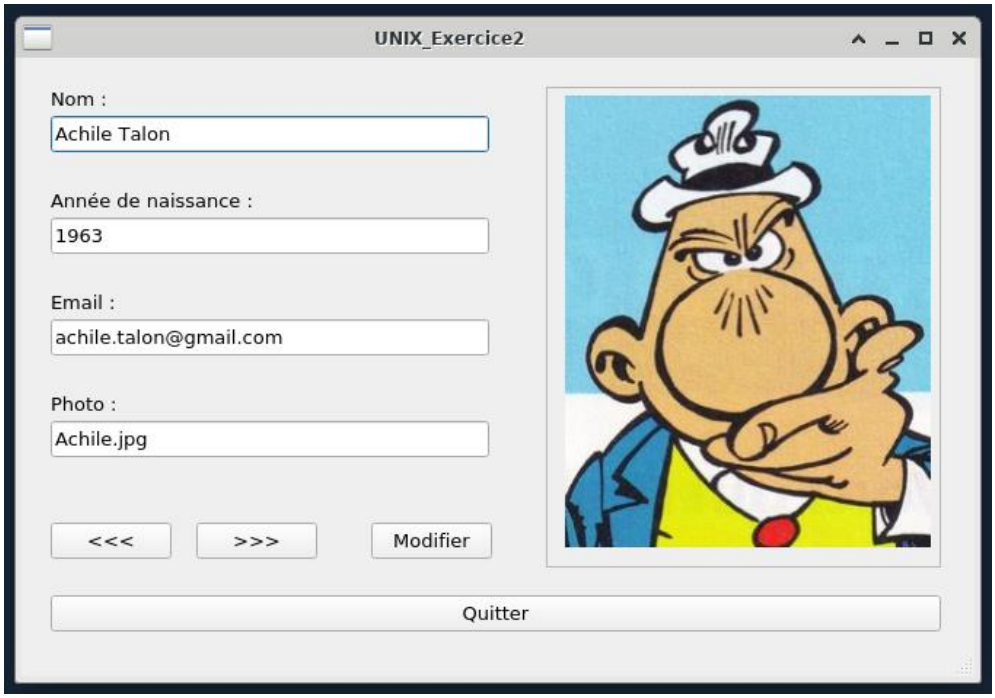
Exercice n°2 : le makefile et les fichiers de bas niveau

Objectifs :

- Se familiariser avec l'utilitaire **make** et la création d'un fichier **makefile**
- Se familiariser avec l'utilisation de la librairie graphique Qt
- Effectuer ses premiers accès **fichiers** de bas niveau (**open**, **close**, **read**, **write** et **lseek**)

Description générale :

L'application ressemble visuellement à



Elle permet d'afficher le nom, l'année de naissance, l'email et la photo des individus gérés par l'application.

La fenêtre graphique a été créée pour vous et vous est fournie sous la forme d'un ensemble de fichiers sources.

Etape 1 : Création d'un fichier makefile

On vous fournit une archive **UNIX_Exercice2.tar** qui contient les fichiers suivants :

- **main.cpp** : le main de votre application
- **mainwindow.cpp** : code source de la fenêtre de l'application, seul fichier que vous devez modifier.
- **mainwindows.h** : fichier header correspondant au fichier mainwindow.cpp.
- **moc_mainwindow.cpp** et **ui_mainwindow.h** : fichiers sources propres à Qt nécessaires à la fenêtre graphique.
- **Compile.sh** : fichier texte (on appelle ça un script) contenant les lignes de compilation nécessaires pour la création du fichier exécutable « UNIX_Exercice2 ».

Il est déjà possible de compiler l'application en utilisant le fichier **Compile.sh** :

```
# sh Compile.sh
# UNIX_Exercice2
```

Le but est donc de créer un fichier **makefile** permettant de réaliser la compilation de tous les fichiers nécessaires à la création de l'exécutable UNIX_Exercice2.

Etape 2 : Programmation des fonctionnalités (EN MÉMOIRE) : version 1

Vous trouverez dans le fichier **mainwindow.cpp** :

```
typedef struct
{
  char nom[40];
  char email[40];
  int  anneeNaissance;
  char photo[40];
} ELEMENT;

ELEMENT Elm[] =
{
  {"Achile Talon","achile.talon@gmail.com",1963,"Achile.jpg"},
  {"Gaston Lagaffe","gaston.lagaffe@gmail.com" , 1957,"Gaston.jpg"},
  {"Lucien","lucien@hepl.be",1979,"Lucien.jpg"},
  {"Robert Bidochon","robert.bidochon@hepl.com", 1977,"Bidochon.jpeg"}
};
```

Le vecteur **Elm** contient l'ensemble des individus gérés par l'application. Aucun ajout/suppression ne sera géré ici. Seule la **modification** sera implémentée. Et pour chaque enregistrement, seul l'**email** et la **photo** pourront être modifiés.

Pour l'instant, on vous demande de modifier le code fourni (uniquement dans le fichier **mainwindow.cpp**) de telle sorte que l'application parcoure, affiche et modifie la variable globale **Elm** (**Donc tout se passe en mémoire actuellement !**). Plus précisément :

- Un clic sur le bouton **« Quitter »** termine l'application → vous devez modifier la fonction
  void MainWindow::on_pushButtonQuitter_clicked()

- Un clic sur le bouton **« >>> »** permet d'afficher l'élément suivant du vecteur **Elm** → vous devez modifier la fonction
  void MainWindow::on_pushButtonSuivant_clicked()

- Un clic sur le bouton **« <<< »** permet d'afficher l'élément précédent du vecteur **Elm** → vous devez modifier la fonction
  void MainWindow::on_pushButtonPrecedent_clicked().

- Un clic sur le bouton **« Modifier »** permet de modifier l'élément en cours dans le vecteur **Elm** → vous devez modifier la fonction
  void MainWindow::on_pushButtonModifier_clicked().

N'oubliez pas de gérer le fait que l'on peut atteindre les « extrémités » du vecteur **Elm** et qu'il ne faut pas en sortir.

Afin de modifier le texte des champs de texte apparaissant dans la fenêtre (nom, année de naissance, email et photo), vous disposez des méthodes

- **void MainWindow::setNom(const char* Text)**
- **void MainWindow::setAnneeNaissance(int annee)**
- **void MainWindow::setEmail(const char* Text)**
- **void MainWindow::setPhoto(const char* Text)** : qui modifie non seulement le champ de texte correspondant au nom de la photo mais également la photo dans la partie droite de la fenêtre. Pour cela, la fonction va rechercher le fichier image (au format jpeg) dans le **répertoire photos** fourni avec le code source.

Pour récupérer les informations de la fenêtre graphique, et plus précisément les données encodées par l'utilisateur dans les champs de texte de la fenêtre, vous disposez des méthodes :

- **const char* MainWindow::getNom()**
- **int MainWindow::getAnneeNaissance()**
- **const char* MainWindow::getEmail()**
- **const char* MainWindow::getPhoto()**

Remarquez que les champs de texte correspondant au nom et à l'année de naissance ne sont pas éditables par l'utilisateur.

Etape 3 : Bidonnage d'un fichier d'enregistrements

On vous demande de créer un programme **Bidon.cpp** indépendant qui va créer le fichier binaire **individus.dat** à partir de la structure ELEMENT (que vous devez donc déclarer dans Bidon.cpp) et écrire dans ce fichier les 4 enregistrements donnés plus haut (ceux qui se trouvent dans le vecteur Elm que vous pouvez également déclarer dans Bidon.cpp pour plus de facilité).

Ce programme doit utiliser les appels système **open**, **write** et **close**.

Une fois mis au point, votre **makefile** doit être adapté pour compiler l'exécutable Bidon en plus de UNIX_Exercice1.

Etape 4 : Programmation des fonctionnalités (GESTION DU FICHIER) : version 2

La variable globale **Elm** disparaît de votre programme. Les fonctionnalités restent les mêmes sauf que les boutons **« >>> »**, **« <<< »**, **« Modifier »** permettent d'accéder et de modifier directement le fichier **individus.dat**. Le fichier pourra être ouvert dans le constructeur de la fenêtre et fermé lors du clic sur le bouton **« Quitter »**.

Vous devez donc à présent utiliser les appels système **open**, **read**, **write**, **lseek** et **close**.