Après la calculatrice, les écluses et les machines à laver… un lecteur audio.

MPV est un lecteur audio vidéo open source. C'est un fork de MPlayer2.
L'adresse du site de MPV est http://mpv.io

Outre l'avantage de pouvoir lire à peu près n'importe quels médias, MPV est scriptable, propose une API et peut être piloté par IPC.
C'est cette dernière fonctionnalité que nous allons utiliser.

MPV peut fonctionner en tâche de fond, et communiquer en JSON à travers un socket (domain socket sous UNIX, named socket sous Windows).
Vous allez donc écrire un front-end audio pour MPV utilisant l'IPC, en C++ dans l'environnement Qt.
Il existe de nombreux exemples d'interface pour MPV (utilisant généralement plutôt l'API), mais la vôtre sera ergonomique et fera preuve d'inventivité et d'originalité !

Caractéristiques :

- Le lecteur est un lecteur audio seulement (pas de vidéo).
- Le logiciel client est potentiellement multi-utilisateurs (plusieurs instances peuvent tourner en même temps et seront synchronisées).
- Il peut lire des fichiers, des listes ou des flux radio.
- Les listes peuvent être des listes arbitraires de fichiers, ou des dossiers qui seront vus comme des listes, conformément aux modes proposés par MPV.
- Il est possible de régler le volume, forcer un mode silencieux et revenir au volume précédent.
- Toutes les commandes classiques de contrôle de lecture sont disponibles (avance ou retour rapide, pause…) ainsi qu'un accès direct à un point temporel, et un affichage du temps (position et temps restant).
- Il est possible de lancer, à tout moment, la lecture d'une piste particulière d'une liste.
- Il affiche les métadonnées de la source en cours de diffusion en direct, de présenter les listes avec leurs métadonnées. On utilisera taglib (librairies pour Ubuntu 14.04, includes et explications joints ; pour les autres plateformes, il sera nécessaire de le télécharger à l'adresse http://taglib.org) pour décoder les tags.
- Le client s'adaptera à l'état du serveur au démarrage.
- Le lecteur devra être disponible en plusieurs langues (à votre choix), et ce réglage sera mémorisé dans des préférences.
- Le serveur mémorisera des radios, morceaux et listes qui seront affichés par chaque client dans une table sélectionnable.
- Le serveur reprendra automatiquement la diffusion au point où il l'avait laissée s'il est interrompu (liste, radio, morceau…) et au même volume, état de mute, de pause, etc.

Question subsidiaire (bonus de 2 points)

- Le logiciel peut afficher les programmes des radios en temps réel s'ils sont disponibles en métadonnées du flux.

Quelques contraintes :

- L'environnement *devra* être C++/QtCreator.
- Chaque fichier *devra* spécifier l'identité des auteurs en entête.
- Le dossier contenant le code *devra* porter également le nom des auteurs.
- La version de Qt utilisée *est* 5.8.
- Le logiciel est portable (je ne demande pas de tester sur d'autres plateformes que la vôtre, mais le code devra être non spécifique).
- Les communications avec MPV sont asynchrones et utilisent QLocalSocket/QJson.
- On n'utilise bien sûr pas Qt Multimedia (on commande en asynchrone le serveur MPV).
- Le bloc de contrôles principaux (lecture, pause, etc.) est une machine à états.
- L'IHM utilisera des widgets spécifiques pour les contrôles (au moins 1, héritant directement de QWidget, avec gestion des événements).

Déroulement :

Le travail sera fait individuellement ou, mieux, en binôme. Dans ce dernier cas, la part de chacun *devra* être décrite.
Un TP (semaine du 10 avril) sera consacré à l'étude préliminaire qui devra être rendue en fin de séance (assurez-vous avant de quitter la salle que les fichiers sont bien sur le serveur).
dans cette étude préliminaire,

- Vous ferez un premier schéma d'IHM, *que vous expliciterez* !
- Vous ferez un descriptif des flux de données, et des échanges asynchrones avec le serveur.

Le travail final sera rendu le jeudi 4 mai à 18h. Vous pourrez y joindre un document explicatif précisant certains points si vous le souhaitez.