Programmation événementielle
Python, événements, callbacks,...

Manuel Munier

Version 10 octobre 2023 (09:04)

Objectifs

L'objectif de ce module est de comprendre le fonctionnement de la programmation par événements, c'est-à-dire de d'écrire un programme (en Python) capable de déclencher l'exécution de telle ou telle fonction *callback* pour réagir à la réception d'un événement.

Consignes du projet

- Le développement se fera en langage Python. Votre code devra impérativement être commenté.
- Vous devrez rédiger un court rapport fournissant la description de votre travail. Il ne s'agit pas de simplement fournir le code commenté ! À vous d'expliquer la logique de vos programmes et ce qu'apporte telle couche par rapport à la précédente. Bref, il vous faudra prendre un peu de recul...
- Une courte présentation ainsi qu'une démo seront exigées en fin de projet.

Travail à réaliser

Cette année ce sera "figure imposée". Les applications "Messenger like" sont faciles à comprendre pour s'initier à la programmation événementielle, mais il existe déjà de trop nombreuses solutions sur Internet. Et trop d'étudiants se contentent se recopier le code trouvé sur Internet sans même chercher à comprendre ce qu'il fait ni comment il le fait... Du coup, pour cette année, l'application à développer sera un explorateur MQTT, ce qui devrait vous permettre d'exprimer toutes vos compétences... et uniquement les vôtres...

1. interface graphique en Tkinter $\rightsquigarrow$ partie "dessin", c'est-à-dire création des widgets et des conteneurs, gestion des placements, etc.
2. interface graphique en TKinter $\rightsquigarrow$ partie événements, c'est-à-dire ajout de foncions callback à certains widgets, fonctions callback qui mettent à jour d'autres widgets, etc.
3. multithreading $\rightsquigarrow$ plusieurs threads (légers) capables de lire et/ou modifier des widgets $\rightsquigarrow$ notion de Timer, gestion des connexions réseau, etc.

Pour vous aider voici quelques éléments d'information quant à l'application que vous devrez programmer :

1. Il existe déjà (et heureusement) un certain nombre d'outils pour le protocole MQTT (ex : MQTT Explorer, MQTT.fx). Sans chercher à les copier entièrement, vous pouvez néanmoins vous en inspirer.
2. Votre application devra au minimum proposer des widgets pour :
   - souscrire à un topic particulier
   - afficher les messages reçus (sur les topics auxquels vous avez souscrit) dans une fenêtre de log avec, pour chaque message, le nom du topic, l'heure de réception, et le contenu du message
   - envoyer un message sur un topic particulier
   - exporter le contenu du log dans un fichier texte et/ou dans un fichier JSON
3. Vous pourrez aussi ajouter "ce qu'il faut" pour pouvoir choisir le broker MQTT auxquel on veut se connecter (adresse IP et numéro de port) sans avoir à modifier le code et à redémarrer l'application.
4. Pour afficher les messages reçus vous pouvez mettre en place des onglets pour séparer les logs selon les topics.

Document rédigé en LaTeX sous Linux.