S.E.R.I.A.N.E.
Systèmes & Mesures

https://seriane.fr/formation

Cours Programmation Système sous Linux avec C / Go / Rust

Objectif

Comparaison des trois langages à travers la réalisation d'applications classiques utilisables sur un terminal en ligne de commande, parmi lesquelles :

- des manipulations sur l'arborescence des fichiers (ersatz de cat, cp, ls, …),
- une interface d'exécution de commandes (un shell Bourne simple),
- des utilitaires clients/serveurs utilisant les protocoles UDP et TCP, etc.

Pourquoi ce nouveau cours ?

Le langage par excellence de l'embarqué et des microcontrôleurs est indubitablement le langage C, car il offre la proximité adéquate avec le matériel et permet d'atteindre de belles performances lorsqu'on ne souhaite pas coder en assembleur.

Mais le langage C draine avec lui son lot de difficultés, inhérentes à sa nature permissive, et il exige donc énormément de rigueur de la part du développeur. Aussi, des alternatives existent.
Il ne s'agit pas de mettre dos à dos des langages qui, par ailleurs, se complètent généralement très bien. Cependant, force est de constater que la permissivité du C semble faire de plus en plus peur, au point que l'on désigne quelquefois le langage comme « non sûr », malgré le fait que, dernièrement, nombre de failles assez catastrophiques concernent des langages de bien plus haut niveau. Quoi qu'il en soit, la tendance est donc à limiter l'utilisation du C (et du C++) pour se tourner vers des langages décrits comme plus robustes. **Deux d'entre eux se partagent la vedette dans ce domaine en ce moment : Rust et Go.** Ils ont d'ailleurs des points communs.

Même si nous sommes de grands amateurs de C, les sirènes de la modernité nous poussent à élargir nos horizons. Peut-être hésiterons-nous entre Rust, qui est sur le point d'intégrer le noyau Linux et Go, qui compte parmi ses géniteurs rien de moins que Ken Thompson (inventeur du langage B, prédécesseur du C de Dennis Ritchie, et co-créateur d'UNIX) et Rob Pike. Il existe des implémentations pour microcontrôleur aussi bien de Rust que de Go, mais le choix de l'un ou l'autre (si tant est qu'il faille choisir) tiendra généralement des préférences personnelles, concernant la syntaxe, l'environnement de compilation, la disponibilité pour une plateforme donnée, etc.

Et les langages de script comme Python ?

Nous parlons ici d'alternatives au C/C++, avec comme base de réflexion le fait d'opter pour quelque chose d'aussi performant en termes de consommation de CPU, mais également de SRAM et de flash. Certes, MicroPython ou CircuitPython permettent de programmer un microcontrôleur et d'obtenir des résultats, mais ces dérivés de Python sont avant tout des outils pédagogiques pour l'apprentissage de la programmation, comme Ardublock, Scratch ou mBlock.

Si des « makers » ou des artistes peuvent se satisfaire de telles solutions et éventuellement changer de plateforme lorsque celle choisie initialement arrive à ses limites, il n'en va pas de même pour des développeurs, d'autant moins en milieu professionnel, pour qui les ressources, comme la mémoire flash, sont précieuses et le cahier des charges non extensible à loisir.

Pour toute information complémentaire : **pfoubet@seriane.org**