Projet « Rock'n'Roll »

Le but du projet est d'implémenter un petit micronoyau et de le faire tourner soit sur une vraie machine soit dans un émulateur. Nous vous conseillons de prendre le micronoyau en OCaml que vous avez vu au premier cours comme spécification. Sinon vous êtes libre d'adapter ou d'étendre cette spécification pour vous concentrer sur un aspect des systèmes qui vous intéresse, comme, par exemple,

— la mémoire virtuelle,
— les systèmes de fichier,
— l'ordonnancement temps réel,
— l'interface réseau, ou
— l'interface avec des périphériques.

Vous avez aussi beaucoup de liberté dans vos choix d'implémentation. Les deux possibilités principales sont

— réimplémenter et étendre le micronoyau en utilisant le langage C pour le faire tourner sur un Raspberry PI, ou dans l'émulateur QEMU ou
— ajouter à QEMU la possibilité de rediriger les appels système vers une fonction OCaml pour exécuter le code original du micronoyau ensemble avec de « vraies »applications binaires compilées écrites en C.

Sinon vous êtes libre de nous proposer d'autres idées. L'important, c'est de maîtriser les détails conceptuels et techniques d'un sous-système important d'un système d'exploitation. Vous pourriez mélanger la modélisation de plus haut niveau, comme des programmes en OCaml, avec la programmation au niveau de bits et d'octets ou vous pourriez vous concentrer seulement sur une implémentation en profondeur.

QEMU

QEMU (« Quick Emulator ») est un émulateur open source qui sait exécuter des programmes ou des systèmes d'exploitation faits pour une machine sur une autre machine. Il peut exécuter des binaires compilés pour une machine sur le système d'exploitation hôte de l'émulateur, c'est-à-dire que les appels systèmes dans l'émulateur sont interceptés, traduits, et renvoyés hors de QEMU sur la machine hôte. Il peut aussi émuler un système complet avec ses périphériques et son propre système d'exploitation. Dans tous les cas le jeu d'instruction émulé est traduit dynamiquement pour exécution directe sur le processeur de l'hôte, ce qui donne de très bonnes performances, mais qui contribue au fait que le code source de QEMU est parfois difficile à comprendre.

Vous trouverez facilement de l'information sur QEMU et son utilisation sur internet. Par exemple, un peu de documentation sur son fonctionnement interne:

`http://qemu.weilnetz.de/qemu-tech.html`,

quelques conseils pour le développement de QEMU:

`http://wiki.qemu.org/Documentation/GettingStartedDevelopers#Getting_to_know_the_code`,

et une page web de Francesco Balducci qui explique la compilation croisée d'un petit programme ARM (« Hello World », bien sûr) et son exécution dans QEMU sous Linux:

`https://balau82.wordpress.com/2010/02/28/hello-world-for-bare-metal-arm-using-qemu/`.

Le code source de QEMU est disponible dans un dépôt git:

```
git clone git://git.qemu-project.org/qemu.git
```

Pour vous donner quelques idées, nous vous suggérons de regarder la fonction `cpu_loop` dans le fichier `linux-user/main.c`, et la fonction `cpu_exec` dans le fichier `cpu-exec.c`.

Modalité de rendu

Le projet est à faire en binôme. Il doit être remis par email à

`Timothy.Bourke@ens.fr` et `Marc.Pouzet@ens.fr`
avant le **20 mai 2018**.

Votre projet doit se présenter sous forme d'une archive `tar` compressée (option `z` de `tar`), appelée `vos_noms.tgz` qui doit contenir un répertoire appelé `vos_noms` (par exemple : `dupont-durand.tgz`). Dans ce répertoire doivent se trouver les sources de votre programme (inutile d'inclure les fichiers compilés). Quand on se place dans ce répertoire, la commande `make` doit compiler votre programme. La commande `make clean` doit effacer tous les fichiers que `make` a engendrés et ne laisser dans le répertoire que les fichiers sources. L'archive doit également contenir un court rapport expliquant les différents choix techniques qui ont été faits et, le cas échéant, les difficultés rencontrées ou les éléments non réalisés. Ce rapport pourra être fourni dans un format ASCII, PostScript ou PDF.