

Systèmes d'exploitation (SYE) *Introduction à SO3*

Profs Daniel Rossier, Alberto Dassatti, Salvatore Valenza
Version 1.0 (2017-2018)

Plan

- Introduction
- Architecture générale
- Environnement de développement
- Structure des projets
- Démarrage de **SO3**

2

Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

Ce chapitre présente l'infrastructure logicielle utilisée dans le cadre des laboratoires SYE.

Les laboratoires sont à réaliser **impérativement** avec la machine virtuelle configurée par l'institut REDS tournant sur *VirtualBox* (matériel fourni au début du semestre).

Introduction (1/2)

- SO3 - *Smart Object Oriented Operating system*
- Système d'exploitation développé par l'institut REDS et Sootech SA (Startup issue du REDS)



- SOO[®] est une marque déposée et se réfère à une technologie brevetée qui permet aux OS (dont SO3) de migrer en *live* d'un dispositif embarqué à un autre.
 - Ces aspects de la technologie ne seront pas considérés dans les labos SYE

3

Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

SO3 est un système d'exploitation léger conçu par l'institut REDS et la société Sootech SA; le système d'exploitation comporte le noyau qui est *Open Source* et des extensions propriétaires permettant de le faire tourner dans un écosystème composé de systèmes embarqués divers implémentant la technologie SOO (*Smart Object Oriented*). Les différents aspects liés à la technologie SOO ne seront pas considérés dans le cadre du cours SYE.

SO3 est écrit en C et assembleur ARM; il s'inspire fortement des concepts modernes d'un système d'exploitation comme *Linux* dont il comprend le même type de *build system*, c-à-d l'ensemble des scripts et des utilitaires nécessaires à la compilation et la production d'images binaires selon une configuration particulière.

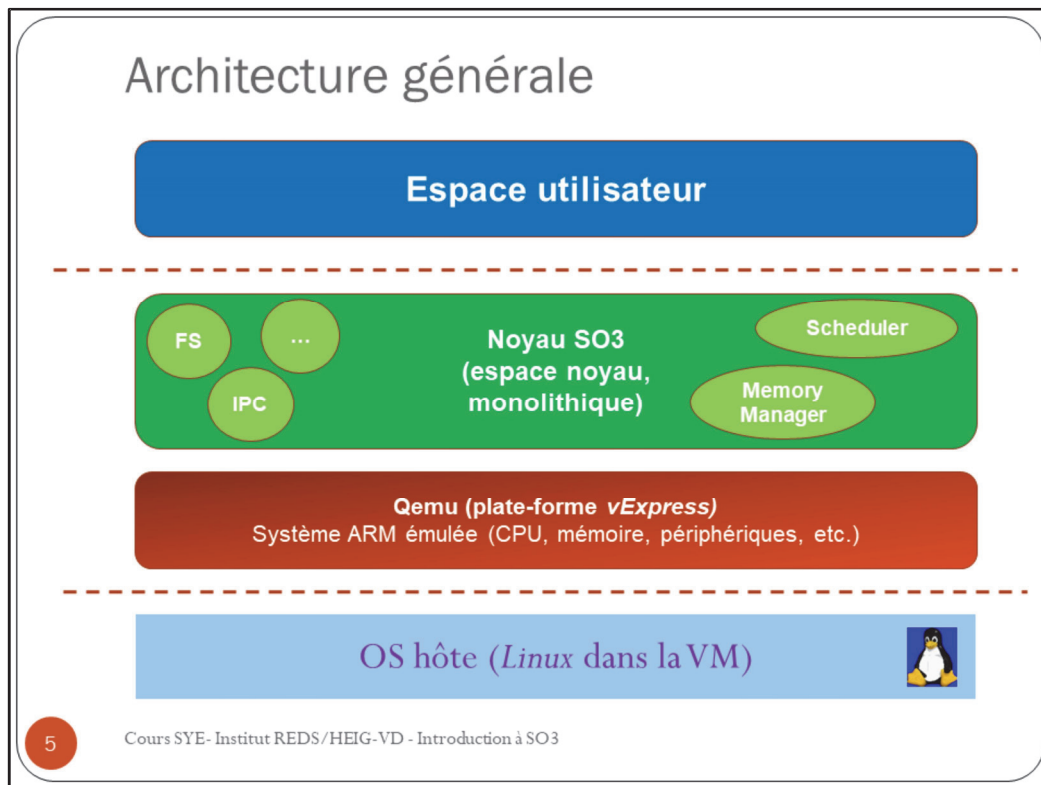
Introduction (2/2)

- Le noyau SO3 est *Open Source* (licence MIT)
- Les extensions SOO de SO3 sont propriétaires
- SO3 est écrit en C (*GNU gcc*) et en assembleur ARM
- Le noyau tourne sur un environnement émulé (*Qemu/vExpress*)
- Grande facilité à *debugger* et à mettre au point



Dans le cadre de ce cours, SO3 tourne sur un environnement émulé basé sur *Qemu* qui émule une plate-forme composée d'un processeur ARM de type *Cortex-A15*.

L'avantage de cette approche réside dans la grande facilité à étudier les entrailles d'un système d'exploitation en disposant d'un environnement de développement ad-hoc. Dans ce sens, le code de SO3 reste abordable du point de vue de l'étudiant et il peut le modifier/l'adapter à sa guise en disposant des outils de mise au point efficaces.

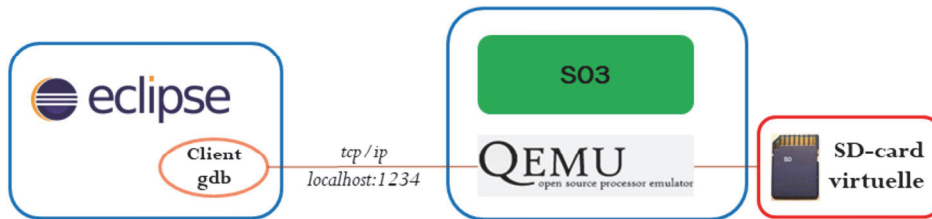


L'architecture présentée ci-dessus montre les trois couches principales de l'environnement:

- **L'espace utilisateur**, qui représente l'ensemble des applications utilisateurs développées en langage C.
- Le **noyau** de l'OS *SO3* comprenant les sous-systèmes principaux que l'on peut trouver dans un OS moderne.
- L'émulateur (*Qemu*) qui émule le **matériel** incluant le processeur et divers périphériques.

Environnement de développement

- Le développement s'effectue avec l'IDE *Eclipse*
 - Edition des fichiers
 - *Debugging*
 - *Métadonnées préconfigurées*
- La compilation s'effectue **en ligne de commande** (*Makefile*)



- Matériel de survie
 - <http://gl.developpez.com/tutoriel/outil/makefile>
 - <http://www.commentcamarche.net/faq/8386-kit-de-survie-linux>

6

Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

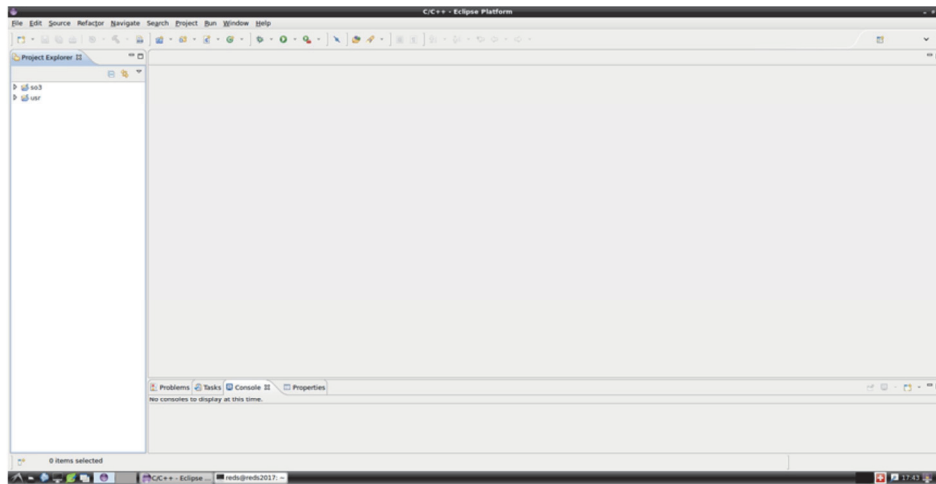
L'émulateur *Qemu* intègre un serveur *gdb* (que l'on peut activer avec l'option `-s` à son démarrage) permettant ainsi à un client *gdb* tel que celui inclus dans l'environnement *Eclipse* de se connecter à une cible émulée (en *remote* via *tcp/ip*). Il devient aisé d'analyser le comportement d'un programme en traçant son exécution en *live*.

Il est bien entendu possible de *debugger* soit le noyau SO3, soit l'application qui tourne sur le noyau. Dans ces deux cas, il s'agit de configurations de *debug* différentes qui ont été préinstallé dans l'IDE *Eclipse*.

L'émulateur est capable d'émuler également un contrôleur de type SD/MMC permettant à l'OS de disposer d'un système de stockage. Ainsi, une carte SD/MMC virtuelle peut être connecté à l'émulateur lors du démarrage de celui-ci (il vaut la peine d'examiner les options du script *stf* qui permet de démarrer SO3).

Structure des projets (1/3)

- Deux projets : **so3** et **usr**



7

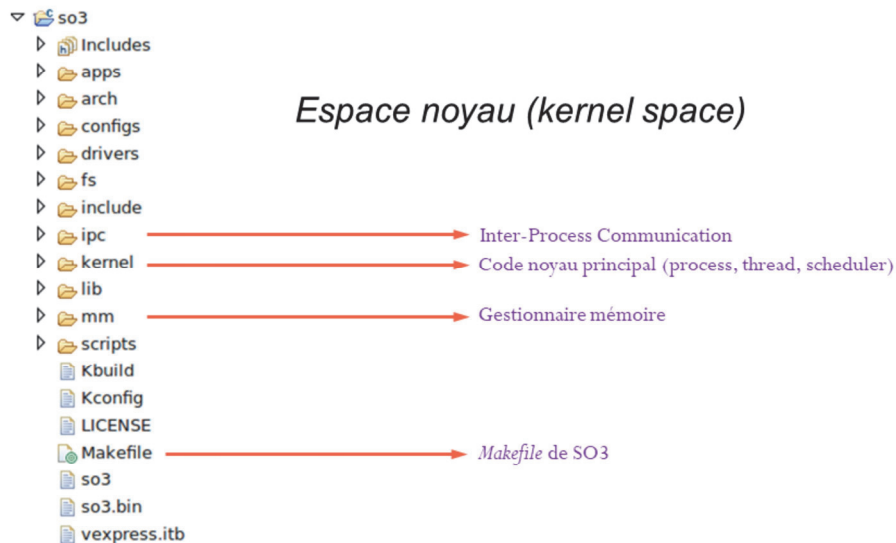
Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

Lors du démarrage d'*Eclipse*, celui-ci a été préconfiguré avec **deux projets** : le projet **so3** et le projet **usr**.

Les deux projets comprennent une arborescence de fichiers et sous-répertoires comprenant l'essentiel du code à étudier et à adapter/compléter.

Le projet *so3* comprend l'ensemble du noyau alors que le projet *usr* comprend les applications *utilisateur* qui tourneront sur SO3. Ce répertoire comprend également la **libc**, librairie fondamentale qui comprend un ensemble de fonctions standards C (par exemple la fonction *printf()*) ou encore les fonctions qui manipulent les chaînes de caractères) ainsi que l'ensemble des **appels systèmes** de l'OS.

Structure des projets (2/3)



8

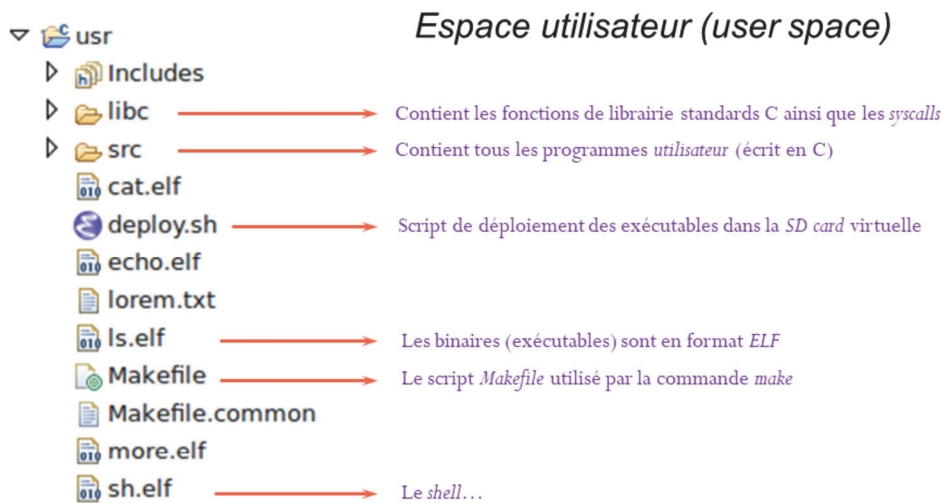
Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

Le répertoire *kernel* de SO3 comprend les sous-systèmes de base tel que la gestion des processus, des *threads* et l'ordonnanceur.

Le répertoire *ipc* contient l'ensemble des mécanismes de communication inter-processus tel que les tubes anonymes.

Le répertoire *fs* contient le code des différents systèmes de fichiers implémentés dans SO3. Ce dernier implémente aujourd'hui principalement le système **FAT-32**, système de fichiers utilisé par la carte SD.

Structure du projet (3/3)



9

Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

Le projet **usr** comprend l'ensemble des applications *utilisateur* et librairies requises par celles-ci.

Ce répertoire contient notamment le *shell* de SO3 qui est démarré comme premier processus (sans parent) par le noyau. Il ne peut donc jamais terminer. Le *shell* permettra ensuite de démarrer d'autres applications en tapant le nom du fichier exécutable correspondant (sans l'extension *.elf*).

Attention ! La compilation des applications (et de la *libc*) s'effectue en ligne de commande et non à partir d'*Eclipse*. Il est nécessaire de faire connaissance avec un minimum de commandes et applications de base dans un environnement *Linux*.

La librairie *libc* dispose de son propre *Makefile* alors que le *Makefile* à la racine est utilisée pour compiler les fichiers se trouvant dans le répertoire *src/*.

Démarrage SO3

```
reds@reds2017: ~/sye/sye_student_2017
File Edit Tabs Help
reds@reds2017:~/sye/sye_student_2017$ ./stf

Starting kernel ...

***** Smart Object Oriented SO3 Operating System *****
Copyright (c) 2014-2017 REDS Institute, HEIG-VD, Yverdon
Copyright (c) 2016-2017 Sootech SA
Version 2017.2.0

Now bootstrapping the kernel ...
SO3: allocating a kernel heap of 1048556 bytes.
memory_init:79 > Found 256 MB of RAM at 0x80000000
SO3: allocating a kernel heap of 1048556 bytes.
memory_init: relocating the device tree from 9fee3000 to 0xc0233000 (size of 4096 bytes)
SO3 Memory information:
- kernel size including frame table is: 2371584 (243000) bytes, 2 MB
- Frame table size is: 64972 bytes

timer_start: setting up system timer periodic freq at 2710
SO3: starting the initial process (shell) ...

so3% █
```

10

Cours SYE- Institut REDS/HEIG-VD - Introduction à SO3

Le démarrage de l'environnement SO3 s'effectue à l'aide du script **stf**.

Il suffit de taper `./stf` dans une terminal, en ligne de commande.

Références

- Profs Daniel Rossier, Alberto Dassatti, Salvatore Valenza. **Cours SYE**