HE-IG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Fiorenzo Gamba, Marina Zapater
Assistants : David Truan, Lucas Elisei, Mattia Gallacchi

Environnement, *shell* et architecture système

lab01 (20.09.2021)

Objectifs de laboratoire

Il s'agit d'un laboratoire d'introduction à l'environnement de travail pour les laboratoires SYE. Dans ce cadre, chaque étudiant-e aura l'occasion de se familiariser avec la compilation du système d'exploitation SO3 et ses différentes applications, d'effectuer différentes manipulations avec *Eclipse* (notamment avec le debugger) ainsi que d'exercer la procédure de rendu.

Etape 1 - Récupération des sources, environnement *Eclipse* et démarrage de SO3

La première étape à effectuer permet de récupérer l'environnement complet nécessaire aux laboratoires SYE.

a) Il est vivement recommandé d'effectuer cette commande dans un dossier « **sye** » qui contiendra tous les laboratoires. La récupération du dépôt s'effectue à l'aide de la commande suivante :

```
reds@reds2021:~$ retrieve_lab sye21 lab01
```

Ceci va créer un dossier « **sye21_lab01** » qui contiendra les fichiers nécessaires au laboratoire.

b) Le répertoire du dépôt « sye21_lab01 » correspond aussi au *workspace* d'*Eclipse*. Avant de démarrer *Eclipse*, il faut déployer les métadonnées (*metadata*) disponibles sur le site *Moodle* afin d'obtenir un *workspace* fonctionnel. Pour cela, depuis un terminal, lancer le script *update.sh* dans le dossier « **sye21_lab01** » comme suit :

```
reds@reds2021:~$ cd sye/sye21_lab01
reds@reds2021:~/sye/sye21_lab01$ ./update.sh
```

c) Ouvrir le *workspace* depuis *Eclipse* et naviguer dans les deux projets (*so3* et *usr*).

⇒ La compilation ne s'effectue pas à partir d'*Eclipse* mais depuis un *shell*, en ligne de commande.

d) La compilation peut s'effectuer depuis la racine du *workspace* (i.e. du dépôt) en tapant la commande *make*. Puis le lancement de *so3* s'effectue à l'aide du script **st** comme ci-dessous

```
reds@reds2021:~/sye/sye21_lab01$ make
reds@reds2021:~/sye/sye21_lab01$ ./st
```

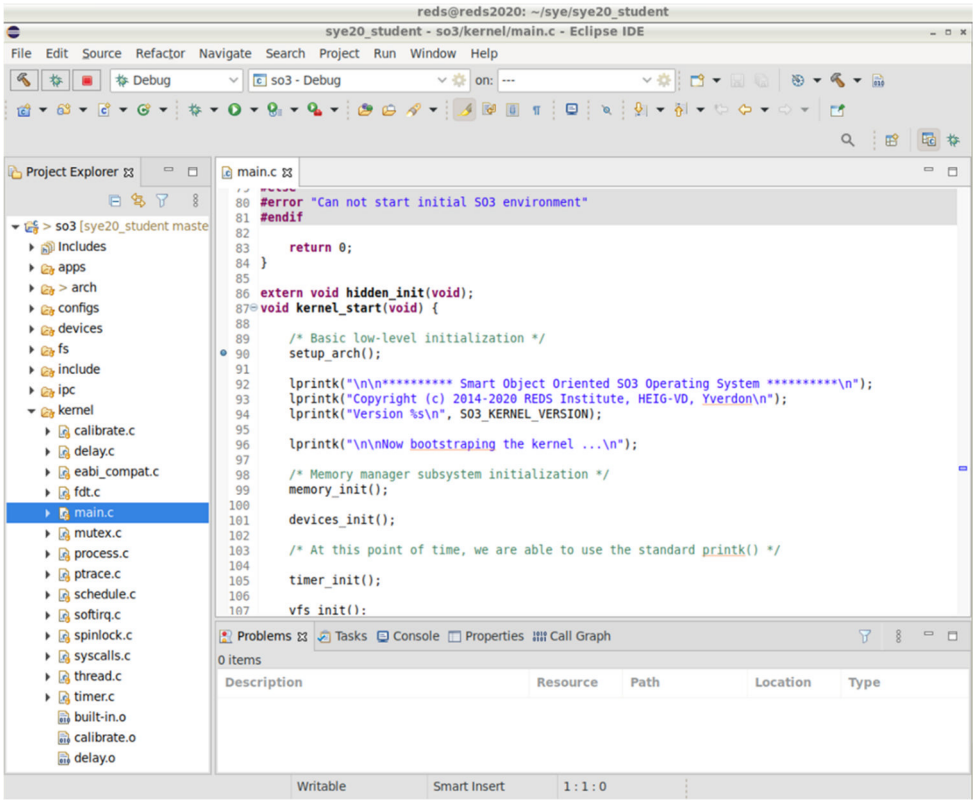
Le script démarre l'émulateur (*QEMU*) qui émule une plate-forme avec un CPU de type ARM 32-bit. L'émulateur commence l'exécution de l'environnement logiciel en démarrant le *bootloader* (*U-boot*) qui lance le noyau SO3. Celui-ci affiche les informations d'initialisation ainsi que le numéro de version du noyau.

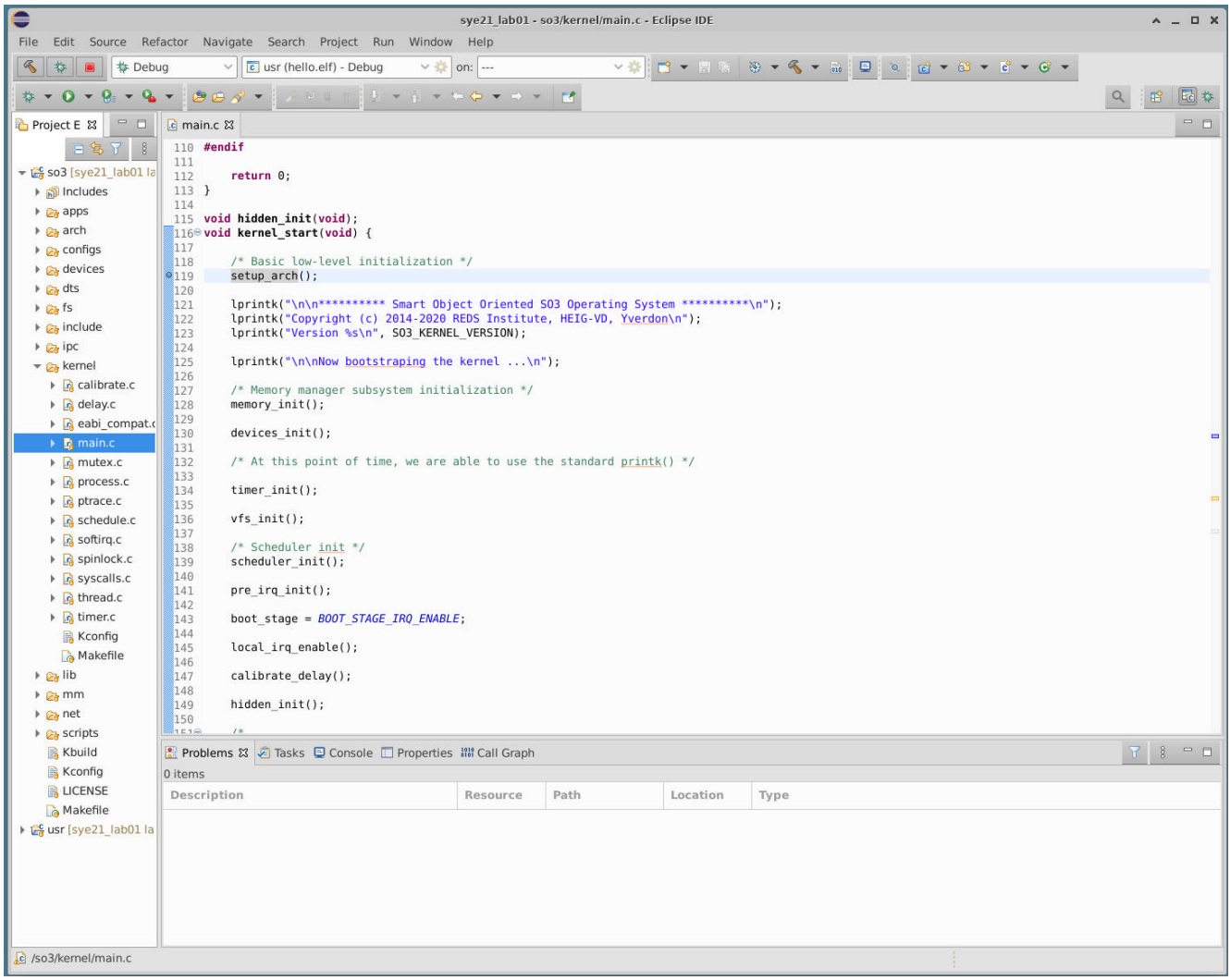
Actuellement, le noyau termine son démarrage par un ... ***"kernel panic"*** précédé par un message d'erreur "*abort exception*". L'étape suivante permettra de corriger le problème. Presser Ctrl+A et après X pour sortir de qemu.

Etape 2 **- Première incursion avec le *debugger* dans *Eclipse*** (espace noyau)

Afin de résoudre le problème précédent, il est proposé de *debugger* l'exécution du noyau avec *gdb* depuis *Eclipse*.

a) Dans *Eclipse*, introduire un *breakpoint* à la ligne 119 dans le fichier so3/kernel/main.c en double-cliquant à gauche du numéro de ligne (voir figure ci-dessous).

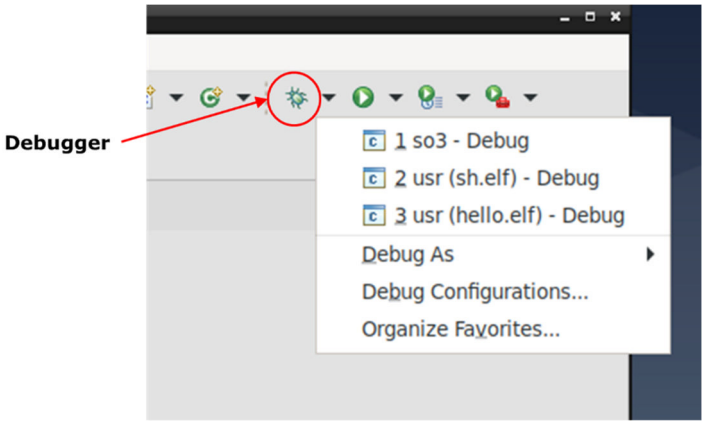




b) Démarrer l'exécution depuis la ligne de commande, en rajoutant l'option **-S** (cette option a pour effet de stopper l'exécution dès le début afin que le client *gdb* ait le temps de se connecter au serveur *gdb*.

```
reds@reds2021:~/sye/sye21_lab01$ ./st -S
```

c) Depuis *Eclipse*, sélectionner la configuration de *Debug* "**so3 - Debug**" tel que montré sur la figure ci-après.



d) L'exécution s'arrête au *breakpoint* (ligne 90). Le nom de la fonction suggère qu'il s'agit de la première fonction C du noyau (le code exécuté auparavant est en assembleur). A partir de là, un appui sur la touche F6 (*step over*) permet d'exécuter la fonction sans y entrer alors que F5 (*step into*) permet de poursuivre l'exécution en entrant dans chaque fonction.
e) Poursuivre l'exécution jusqu'au problème d'exécution, puis corriger le code (il suffira de supprimer l'appel de la fonction posant problème). Ne pas oublier de recompiler !
f) Redémarrer SO3 et s'assurer du bon fonctionnement.

A ce stade, le noyau démarre la première application du système - le ***shell*** - qui affiche une invite (*prompt*) sous la forme « *so3%* ». L'utilisateur peut alors entrer des commandes et lancer des applications.

L'application « ***ls*** » permet d'afficher la liste des entrées de répertoire du répertoire courant.

e) Taper la commande « ***ls*** » et constater qu'une erreur survient.

⇒ Pour quitter *QEMU*, il faut taper « **ctrl+a** » puis « **x** » (en relâchant « ctrl+a »).

Etape 3 **- Utilisation du *debugger* pour une application** (espace utilisateur)

L'espace utilisateur définit l'ensemble du code qui s'exécute lorsque le processeur fonctionne en mode « *user* ». Cet espace comprend l'ensemble des applications de type *utilisateur*.
Afin de résoudre le problème précédent, il est proposé de *debugger* l'exécution de l'application (dans l'espace utilisateur) avec *gdb* depuis *Eclipse*.

a) Lancer le script de démarrage (./**st**), cette fois-ci sans l'option -S. En effet, le problème survenant après l'invite de commande du *shell*, il n'est pas nécessaire que le client se connecte dès le début de l'exécution.
b) Depuis *Eclipse*, introduire un *breakpoint* à la ligne 255 (soit juste après l'entrée de l'utilisateur) dans le fichier « `usr/src/sh.c` »
c) Depuis *Eclipse*, sélectionner la configuration de *Debug* "**usr - Debug**" (*sh* est le nom de l'application correspondant au *shell*). Effectuer du pas-à-pas jusqu'à ce que le problème survienne.
d) Corriger le problème et s'assurer que tout fonctionne.

Etape 4 - Modification d'octet (*byte*) en mémoire

Le langage C permet d'accéder la mémoire au *byte* prêt. Dans cette étape, il s'agit de modifier le contenu d'une chaîne de caractères *byte* par *byte*.

⇒ (Petite précision) Les manipulations suivantes sont possibles dans l'environnement SO3, car le noyau ne charge pas les zones mémoires à des emplacements aléatoires au chargement d'une application, ce qui est le cas sous *Linux* avec une configuration standard.

a) Editer le fichier « *usr/src/hello.c* » et compléter la fonction « *main()* » afin d'afficher une chaîne de caractères avec le contenu suivant : "Hello world !"
b) Compléter le programme afin d'afficher l'adresse de la chaîne de caractères.
c) Compléter le programme afin d'incrémenter la valeur de chaque *byte* de cette chaîne de caractères en passant par **l'adresse** de la chaîne (en valeur numérique à l'aide des bons « *cast* » et non par sa variable). L'adresse reste la même durant les executions car SO3 ne reloge pas son code.

Etape 5 - Modification du *shell*

L'espace utilisateur définit l'ensemble du code qui s'exécute lorsque le processeur fonctionne en mode « *user* ». Cet espace comprend l'ensemble des applications de type *utilisateur*. Une application de base est le *shell* évoqué précédemment.

Cette étape consiste à modifier le *shell* afin que celui-ci affiche la liste des *tokens* à chaque entrée d'une commande. Par *token*, on entend chaque élément de la chaîne de caractères (entrée par l'utilisateur) séparée par un espace.

a) Editer le fichier « *usr/src/sh.c* » et analyser le code de la fonction principale (« *main()* »).
b) Compléter le fichier afin d'afficher la liste des *tokens* après chaque entrée de commande. L'affichage se présentera comme les exemples ci-dessous :

```
so3% echo a bc de
echo
a
bc
de
a bc de
so3% ls
ls
cat.elf
dev/
echo.elf
lorem.txt
ls.elf
more.elf
sh.elf
so3%
```

Liste des *tokens*

Résultat de l'application

Token (unique)

⇒ Afin de bien visualiser les erreurs de compilation, il est préférable d'effectuer la commande « *make* » dans le répertoire « ***usr/*** » avant de lancer *make* dans le répertoire principal. Il est de même pour la compilation du noyau (répertoire « ***so3/*** »).
⇒ Afin d'afficher les lignes sur le terminal, il ne faut pas oublier de terminer les chaînes de caractères par un retour à la ligne '\n'.

Etape 6 - Modification de la version du noyau

La version actuelle du noyau est le numéro **2021.4.1**. Afin de poursuivre l'exploration du noyau, il est proposé de changer la version du noyau à **2021.4.2**. Pour rappel, le numéro de version est affiché au démarrage du noyau.

a) Editer le fichier du noyau contenant le numéro de version et définir la version **2021.4.2**.
b) Compiler et vérifier le numéro de version à l'affichage.

⇒ L'utilisation de l'outil **grep** de la suite GNU permet la recherche d'un *pattern* dans les fichiers spécifiés. La commande ci-dessous va afficher toutes les occurrences de « toto » présentes dans tous les fichiers du dossier courant et des sous-répertoires (fouille récursive).

```
reds@reds2021:~/sye21_lab01$ grep -r toto
```

⇒ Les pages « ***man*** » fournissent de l'aide sur les commandes ; voici un exemple d'utilisation :

```
reds@reds2021:~/sye21_lab01$ man grep
```

Etape 7 - Modification de la routine de service (ISR) du *timer*

Dans SO3, le *timer* système est programmé à une fréquence de 1000 Hz. Autrement dit, une interruption *timer* de type IRQ est générée au processeur toutes les millisecondes. Pour information, le *timer* de la plate-forme émulée *vExpress* est de type « *sp804* ».

a) Démarrer SO3 ainsi que le *debugger* au niveau du noyau.
b) Insérer un *breakpoint* à l'entrée de la routine de service (ISR) du *timer* dans le fichier « *so3/devices/timer/sp804.c* ».
c) A gauche de l'environnement, examiner le chemin d'appel des fonctions (*backtrace*) au moment de l'arrêt du programme sur le *breakpoint*.
d) Modifier la routine de service afin d'afficher toutes les secondes un compteur qui donne le nombre d'interruptions.

⇒ Dans le noyau, la fonction « *printf()* » est remplacée par la fonction « ***printk()*** ».

Etape 8 - Validation du laboratoire

Selon la méthode énoncée en début de semestre, chaque fin de session doit aboutir à l'exécution d'un script qui transmettra le travail fourni. Chaque laboratoire s'accompagne d'un script de rendu appelé *rendu.sh* qu'il est **nécessaire** d'appeler à la fin de chaque session :

```
reds@reds2021:~/sye/sye21_lab01$ ./rendu.sh session
```

Cette commande va générer un dossier *rendu* qui contiendra les différences avec le dépôt de base.

Lorsque le laboratoire est terminé - en principe à la fin de la dernière session du labo - il suffit **d'exécuter le script** en lui spécifiant **fin** comme paramètre :

```
reds@reds2021:~/sye/sye21_lab01$ ./rendu.sh fin
```

Cette commande va générer un dernier fichier de différences, puis archiver le dossier *rendu*. Il est demandé de déposer cette archive (*rendu.tar.gz*) dans Moodle à la fin du laboratoire.