HEIG-VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Debug kernel/user, application user et mémoire

lab02 (Semaine du 25.09.2023)

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 *VSCode* (notamment avec le debugger) ainsi que d'exercer la procédure de rendu.

Validation du laboratoire

Ce laboratoire sera à rendre selon la méthode vue lors du précédent laboratoire, à savoir

1. Ajouter les fichiers modifiés avec "*git add <vos_fichiers>*"
2. Commiter les changements avec "*git commit –m "<message de commit>"*
3. Envoyer les changements sur votre repo gitlab avec "*git push –u origin*"

Etape 1 - Environnement VSCode et démarrage de SO3

La première étape à effectuer permet de récupérer la nouvelle branche

a) Récupérer la nouvelle branche lab02 avec la commande suivante:

```
reds@reds2023:~/sye23_student$ git fetch upstream
reds@reds2023:~/sye23_student$ git checkout lab02
```

b) 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@reds2023:~/sye23_student$ make
reds@reds2023:~/sye23_student$ ./st
```

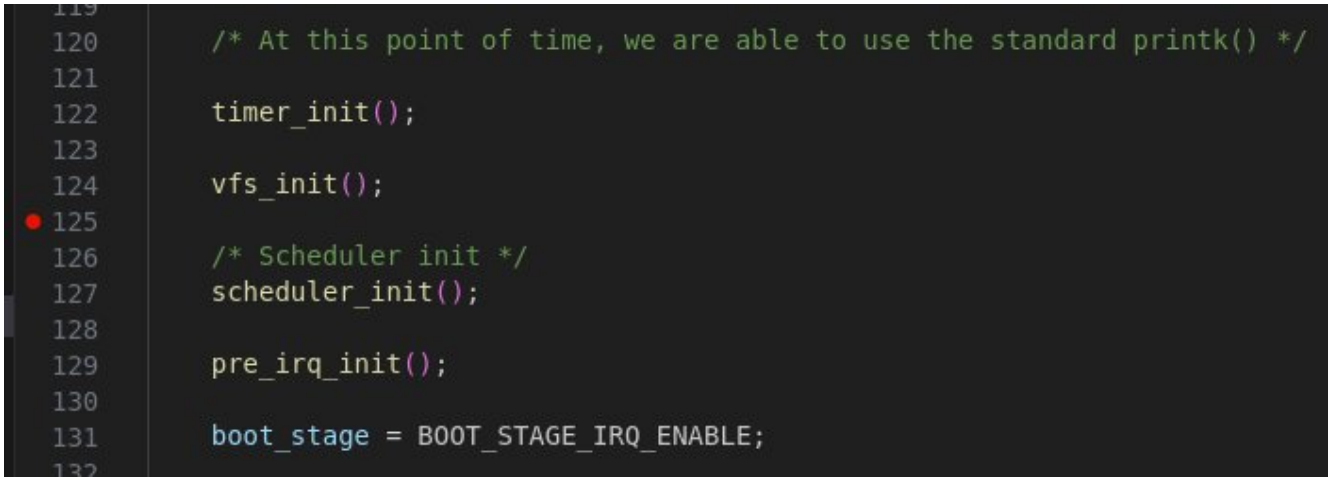
Le script **st** démarre l'émulateur (*QEMU*) qui émule une plate-forme avec un CPU de type ARM 32-bit. L'émulateur commence l'exécution de l'environnement logiciel en démarrant le *bootloader* (*U-boot*) qui lance le noyau SO3. Celui-ci affiche les informations d'initialisation ainsi que le numéro de 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.
⇒ Pour quitter *QEMU*, il faut taper « **ctrl+a** » puis « **x** » (en relâchant « ctrl+a »).

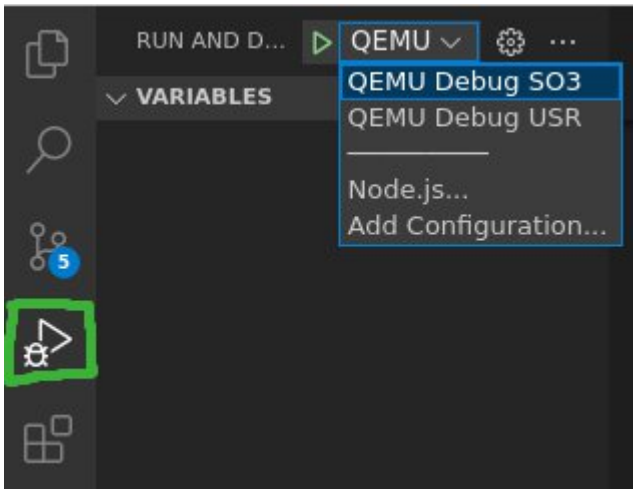
Etape 2 - Première incursion avec le *debugger* (espace noyau)

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

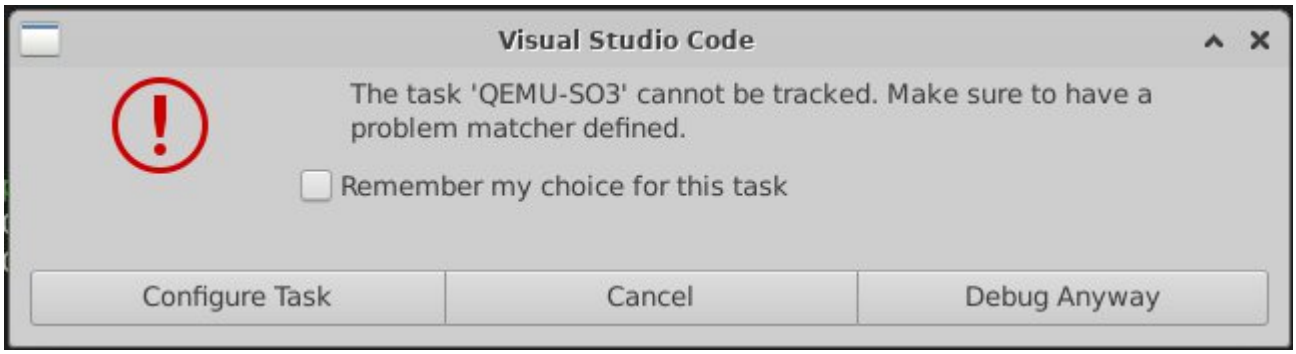
a) Dans *VSCode*, introduire un *breakpoint* à la ligne 125 dans le fichier so3/kernel/main.c en cliquant à gauche du numéro de ligne (voir figure ci-dessous).



b) Dans l'onglet *Run and Debug*, sélectionner *QEMU Debug SO3* dans le menu déroulant en haut de la fenêtre, puis cliquer sur la flèche verte à gauche.



Si vous obtenez une fenêtre d'erreur de VSCode comme sur la figure suivante



Vous pouvez cocher *Remember my choice for this task* puis cliquer sur *Debug Anyway*

c) L'exécution s'arrête au *breakpoint* (ligne 125). 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 F10 (*step over*) permet d'exécuter la fonction sans y entrer alors que F11 (*step into*) permet de poursuivre l'exécution en entrant dans chaque fonction. Vous pouvez également utiliser le petit menu apparu en haut de l'écran pour effectuer les mêmes actions.

d) Poursuivre l'exécution jusqu'au problème d'exécution, puis corriger le code (il suffira de commenter l'appel de la fonction posant problème). L'état de l'exécution peut être observé dans l'onglet "Terminal". Ne pas oublier de recompiler !
e) 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.

c) 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 VScode.

a) Introduire un *breakpoint* à la ligne 198 (soit juste après l'entrée de l'utilisateur) dans le fichier « usr/src/sh.c »
b) Sélectionner la configuration *QEMU Debug USR* puis cliquer sur la flèche verte. Dans la liste qui s'ouvre, sélectionner l'application userspace à débuguer (ici *sh*)
c) 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) Créer le fichier « *usr/src/hello.c* » et ajouter une fonction « *main()* » permettant d'afficher une chaîne de caractères avec le contenu suivant : "Hello world !"
b) Afin d'ajouter le programme *hello* comme commande utilisable dans le terminal SO3, modifiez le fichier "*usr/src/CMakeLists.txt*" en vous inspirant du contenu déjà présent
c) Compléter le programme afin d'afficher l'adresse de la chaîne de caractères.
d) A la suite du programme déjà écrit, utiliser l'adresse affichée lors de l'exécution précédente pour accéder à la chaine de caractère dans le code et en modifier le contenu byte par byte en incrémentant chaque byte de 1 avant de la ré-afficher. **Utiliser l'adresse en dur (constante littérale)** avec les bons "cast" et non **pas en la stockant dans un pointeur**. Cette manipulation est possible car SO3 loge toujours le programme au même endroit et les adresses sont donc toujours les même entre deux exécutions