M2

Portage de MutekH/Hexo sur les processeurs x86-64 et Sparc

Error: Macro Include(StageContexte/SoclibMutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est double.

L'étudiant commencera par le portage de l'exo-noyau Hexo sur le processeur x86 64 bits de type CISC. Hexo et MutekH s'exécute déjà nativement sur l'architecture intel x86 32 bits classique des PCs.

Ce stage implique la prise en main du code du projet MutekH et plus particulièrement de l'hexo noyau Hexo et son portage x86. Le stagiaire devra développer les routines en assembleurs et en C des différents services : démarrage du système, handler d'interruption, changement de contexte, mémoire virtuelle, etc. Ce code pourra s'inspirer fortement du portage 32 bits existant. L'exécution du système sur une machine munie d'un processeur 64 bits permettra de valider cette partie du stage.

Une fois ce premier portage réalisé, l'étudiant aura une bonne connaissance du fonctionnement et de l'arhitecture de l'exo-noyau Hexo. Il pourra alors réaliser le portage sur processeur Sparc V8. Pour cette partie, la validation sera réalisée sur la plate forme de simulation SoCLib. D'autres processeurs de type RISC similaires sont déjà supportés par MutekH et seront une source d'inspiration pour l'implémentation, mais le processeur Sparc présente une réelle particularité qui ouvre des pistes intéressantes quant à l'optimisation du changement de contexte. Le stagiaire pourra alors donner une certaine dimension recherche à ce stage.

Encadrement

L'encadrement de ce stage sera effectué par Dimitri Refauvelet.

Ajout du support du système de fichiers en réseau NFS dans MutekH

Error: Macro Include(StageContexte/MutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est le développement d'un driver de système de fichier Network File System (NFS) pour la nouvelle couche de système de fichiers virtuel de MutekH. Ce driver permettra à MutekH d'accéder aux fichiers au travers du réseau et viendra compléter la collection de systèmes de fichiers déjà supportés : Fat, Iso9660, RamFS...

Le stagiaire devra prendre en main le projet MutekH; plus particulièrement la pile réseau (libnetwork) et la gestion du système de fichiers (libvfs) pour développer le driver. Un serveur de fichiers NFS sera mis en place, l'accès aux fichiers depuis une application tournant sur MutekH permettra de valider le stage.

Encadrement

L'encadrement de ce stage sera effectué par Nicolas Pouillon et Joel Porquet.

Elaboration d'un composant réseau pour SoCLib et développement de son driver pour MutekH

Error: Macro Include(StageContexte/SoclibMutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est double. Tout d'abord il s'agit de développer un modèle de périphérique réseau pour la plate forme de prototypage virtuelle SoCLib. Ceci permettra à une simulation tournant sous SocLib? d'accéder au réseau de la machine hôte. Le modèle de simulation utilisera les mécanisme de DMA classiques qui sont déjà employés par d'autres composants. Il exploitera également la technologie TUN/TAP qui permet de créer facilement une interface réseau virtuelle sur la machine hôte. Ceci permettra d'immerger la simulation dans un vrai réseau, les paquets réseau entrant par l'interface du noyau Linux seront reçus par le périphérique SoCLib et inversement.

Le second objectif est le développement du driver associé pour piloter l'interface réseau depuis MutekH qui contient déjà la pile réseau nécessaire et des pilotes pour certains périphériques existants, MutekH étant déjà capable d'accéder au réseau lorsqu'il s'exécute sur certaines plateformes dont les PCs x86.

Le stage sera validé en exécutant une application sous MutekH capable d'échanger des paquets avec le réseau de la machine hôte.

Ajout du support pour le système de fichiers Ext2 dans MutekH

Error: Macro Include(StageContexte/MutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est le développement d'un driver pour la famille des systèmes de fichiers Ext2, Ext3 et Ext4. Ces systèmes de fichiers sont ceux de base généralement employés par les systèmes d'exploitation GNU/linux.

Le support de ces systèmes s'appuiera sur la bibliothèque existante Ext2fs qui contient l'algorithmique complexe nécessaire. Ce driver permettra à MutekH d'accéder, par exemple, aux fichiers d'une partition GNU/Linux et viendra compléter la collection de systèmes de fichiers déjà supportés : Fat, Iso9660, RamFS...

Le stagiaire devra prendre en main le projet MutekH et notamment la gestion du système de fichiers (libvfs) ainsi que la couche d'accès aux périphériques de blocs pour développer le driver. Il devra également prendre en main et adapter la bibliothèque Ext2fs qui est utilisée pour l'accès aux systèmes de fichiers de cette famille sous GNU/Linux.

Encadrement

L'encadrement de ce stage sera effectué par Nicolas Pouillon et Joel Porquet.

Elaboration d'un modèle de crypto-processeur pour SoCLib et ajout du support dans MutekH

Error: Macro Include(StageContexte/SoclibMutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est le développement d'un modèle de composant SoCLib de type crypto-processeur et de son driver de périphérique pour MutekH.

Les crypto-processeurs sont des périphériques capables de chiffrer et de déchiffrer des données en mémoire par des opérations DMA. Les algorithmes de chiffrement sont un élément clef de la sécurité et s'implémentent particulièrement bien en matériel. Ces crypto-processeurs permettent de soulager les processeurs généralistes pour les opérations cryptographiques et sont de ce fait très employés dans les SoC.

Le stagiaire devra définir le jeu de registre et les automates du nouveau périphérique avant de réaliser son implémentation. Il s'agit d'étudier les mécanismes (DMA, IRQ, ...) à implémenter pour permettre la collaboration entre le logiciel et le matériel qui s'échangeront les données à traiter. Parallèlement, le développement du driver de périphérique pour MutekH et son intégration à la libcrypto qui gère déjà des algorithmes logiciels permettra de tester le modèle de composant.

Le stage sera validé en appliquant une série d'opérations cryptographiques à un jeu de données et en comparant les résultats obtenus entre les implémentations matérielles et logicielles d'un même algorithme. Ce stage n'implique pas nécessairement le développement ou la compréhension profonde des différents algorithmes cryptographiques, dont les implémentations existent déjà sous forme de code libre.

Encadrement

L'encadrement de ce stage sera effectué par Alexandre Becoulet et Geoffrey Plouviez.

Développement d'un driver de MMU pour l'ARM9 et support de la console de jeu GP32 dans MutekH

Error: Macro Include(StageContexte/MutekH) failed

```
pre-0.11 Wiki macro Include by provider <class 'includemacro.macros.IncludeMacro'> no longer sup
```

Objectif

L'objectif de ce stage est d'ajouter à l'exo-noyau Hexo le support de la mémoire virtuelle pour le processeur ARM employé dans la console de jeu Gp32.

Une MMU(Memory Management Unit) est un composant matérielle qui traduit les adresses mémoires employées par le code qui s'exécute en adresses physiques à destination des composants mémoire. En effet, la MMU permet la mise en oeuvre de la mémoire virtuelle où l'espace d'adressage virtuel et physique sont découpés en pages qui sont permutées. La mémoire virtuelle est la base de tous les systèmes d'exploitations modernes multi-utilisateur et sécurisés. Elle rend possible la protection de la mémoire, la séparation des processus en mémoire, la mémoire partagée, le swapping, la création efficace de processus par copy-on-write, et bien d'autre mécanismes essentiels de l'OS... Depuis longtemps employé dans les stations de travails et dans les serveurs, elle l'est aujourd'hui de plus en plus également dans l'embarqué.

Suivant les processeurs et les architectures les MMU diffèrent. Les caratéristiques, les fonctionnalités proposées et la manière d'y accéder ne sont pas les mêmes. Par exemple les tailles de pages peuvent être différentes d'une architecture à l'autre et les mécanisme d'accès à la table de page peuvent être logiciels ou matériels.

La gestion de la mémoire virtuelle est décomposable en deux parties dans Hexo/MutekH:

- un driver, spécifique à chaque MMU, effectue des opération bas niveau pour lire/modifier les tables de pages dans Hexo. L'ensemble des drivers de MMU partage une API générique.
- des gestionnaires de page physiques et virtuelles dans MutekH font directement appel au driver via l'API générique.

Pour réaliser le travail l'étudiant devra étudier la documentation de l'architecture ARM et implémenter le driver pour la MMU. Ce driver devra respecter l'API citée précédemment. Hexo s'exécute déjà nativement sur des architectures employant une MMU. Le processeur ARM et la MMU générique du projet SoCLib étant supporté, il sera possible de s'inspirer de ce driver pour effectuer le travail demandé.

La validation se fera soit sur une carte de développement ?SAM9-L9260 d'Olimex, soit sur la console de jeux portable ?GamePark32. Elles utilisent toutes les deux un processeur du type ARM 9 avec MMU. La validation consistera à exécuter une application sur MutekH/Hexo utilisant la mémoire virtuelle.

Encadrement

L'encadrement de ce stage sera effectué par Dimitri Refauvelet.