Fiche descriptive de projet Hub
Rust OS Life

Contexte et but du projet
Le but du projet est d'apprendre comment un système d'exploitation fonctionne, en créant un système d'exploitation qui permette de faire tourner des programmes simples, avec des librairies d'appels systèmes créées pour l'occasion. Un deuxième but est de faire une montée en compétence poussée en Rust.

Porteur(s) du projet
Amoz Pay (amoz.pay@epitech.eu) - Chef de projet - Développeur
Lilian Giroire (lilian.giroire@epitech.eu) - Développeur

Objectif fonctionnel
Créer un Game of Life qui tourne sur le système d'exploitation qui aura été développé. Le Game of Life est un programme simple et autonome, suivant un ensemble de règles qui permettent à un dessin de base d'évoluer dans le temps.

Pour cela, l'OS devra pouvoir gérer un minimum d'aspects I/O: Output sur un écran, et input de paramètres pour le lancement du programme via un clavier. Le système devra pouvoir gérer l'horloge interne, puisque le jeu doit pouvoir évoluer dans le temps. Les utilisateurs futur doivent pouvoir développer leurs propres programmes sur le système; celui-ci doit donc aussi implémenter d'autres fonctionnalités de bases d'un système d'exploitation.

Environnement technique / technologique
QEMU pour développer sans os
Rust avec librairie Core
Mac Mini 2008

Description du livrable
Le livrable sera
- un ordinateur qui tourne sous l'OS réalisé
- Des librairies réalisées pour l'OS
- Un game of life qui utilise la librairie, en kernel space.

Organisation et temporalité
Outils:
Mac mini 2008 + power supply
Clavier (qwerty de préférence)
Ecran

Temps estimé:

- Montée en compétence Rust (création d'un game of life normal et autres): 2 JEH
- Montée en compétence QEMU Compilation et lancement programmes – 0.5 JEH
- Configuration QEMU pour matcher l'ordinateur ciblé: Mac mini 2008 – 0.5 JEH
- Création ou utilisation d'un boot system mutltiboot – 1 JEH
- Management de la Stack – 0.5 JEH
- Fonctions d'affichage / management de la console – 1 JEH
- Implémentation Apic / Timer (Permet les interruptions) - 2 JEH
- Implémentation interruption hardware clavier AZERTY et QWERTY – 1 JEH
- Création mémoire virtuelle – 2 JEH
- Création fonctions d'affichage vectoriel – 3 JEH
- Gestion des block devices – 2 JEH
- Implémentation système de fichiers FAT – 2 JEH
- Création d'un système de modules kernel – 1 JEH
- Création du jeu Game of Life – 1 JEH

Répartition des rôles:
Nous allons travailler en collaboration sur virtuellement toutes les tâches, afin que nous puissions tous les deux apprendre l'ensemble des notions à apprendre. Cependant, voici quelques responsabilités que nous aurons:
Lilian: Responsable Produit - s'assurer que les features développées permettent d'atteindre l'objectif final de pouvoir réaliser le jeu
Amoz: Responsable Faisabilité: s'assurer que les features nécessaires soient bien réalisables techniquement

Total: 19.5

Estimation du projet

Nous estimons à 40 **XP** le projet.

En effet, ce projet est extrêmement pédagogique car il permet de comprendre de nombreux composant hardware, et comment ces ressources physiques sont abstraites pour l'utilisateur.
Beaucoup de recherches sont donc nécessaires, c'est donc un investissement de temps important.
De plus, la complexité du projet dépasse celle de programmes simples, car les bases sur lesquelles ce projet est construit sont nulle: le but est de tout faire de 0.

Etape du projet

Décortiquer les différentes étapes de votre projet en release. La release 1.0 correspond à 25% de votre projet, la release 2.0 à 50% de votre projet, la release 3.0 à 75% de votre projet et la release 4.0 à 100% des attendus de votre projet.

- Release 1.0 : OS avec affichage texte défini à la compilation

    - Création ou utilisation d'un boot system mutltiboot
    - Management de la Stack
    - Fonctions d'affichage / management de la console

- Release 2.0 : Ajout interruptions hardware et affichage vectoriel
    - ⊄ Implémentation Apic / Timer (Permet les interruptions)
    - ⊄ Implémentation interruption hardware clavier AZERTY et QWERTY
    - ⊄ Création mémoire virtuelle
    - ⊄ Création fonctions d'affichage vectoriel
- Release 3.0 : Ajout acces block devices et système de fichiers
    - Gestion des block devices
    - Implémentation système de fichiers FAT
    - Création d'un système de modules kernel
- Release 4.0 : Game of life en module kernel
    - Création du jeu Game of Life