Laboratoire 02 du cours LOG710

- **Chargé de laboratoire :**

  Bali Ahmed, cc-ahmed.bali@etsmtl.ca

- Site web du cours sur Moodle:

https://ena.etsmtl.ca/course/view.php?id=8734

Laboratoires (Titres)

- **Conception et implémentation d'un interprète de commandes**
  - **Période : 04 semaines**
  - **Pondération : 10 %**

- **Conception et implémentation d'appel système dans le noyau Linux**
  - **Période : 04 semaines**
  - **Pondération : 10%**

- **Conception, implémentation et test d'une API simplifiée d'un gestionnaire de mémoire avec plusieurs stratégies d'allocation de mémoire**
  - **Période : 04 semaines**
  - **Pondération : 10 %**

LAB-02 : Conception et implémentation d'appel système

- **Première partie : Processus de configuration, compilation et installation du noyau Linux**

  - Vous allez travailler sur une machine virtuelle sur laquelle est déployée une distribution Ubuntu.

LAB-02 : Première partie

- **Préparation de l'environnement :**
  - 1) Téléchargement du code source du Kernel :

    sudo apt-get update

    sudo apt-get install linux-source libncurses5-dev
  - 2) Pour visualiser le code source téléchargé

    ls /usr/src/
  - 3) Version actuel du kernel

    uname –r
  - 4) Décompression du code source téléchargé

    tar xjvf /usr/src/linux-source-3.13.0.tar.bz2

LAB-02 : Première partie

- **Compilation :**
  - 5) Configuration

    ```
    cd /boot/linux-source-3.13.0
    cp /boot/config-`uname -r` .config
    make oldconfig
    ```

  - 6) changer la variable EXTRAVERSION du fichier Makefile (initialiser cette option avec la chaine -log710e2016XXYYZZ où XX,YY, ZZ initials des membre del'équipe)
  - 7) Compilation du kernel : Vous pouvez utiliser "parallel make" via make -j en mettant le nombre de cœurs du processeur + 1 (e.g. 3 pour un processeur "dual core")

    ```
    make -j3
    ```

  - 8) Compilation des module

    ```
    make modules -j3
    ```

LAB-02 : Première partie

- **Installation :**
  - 9) Installation des modules : Création des fichiers des modules du noyau dans un répertoire spécifique à la version de votre noyau au niveau du répertoire **/lib/modules**

    make modules_install -j3
  - 10) Installation du kernel : Création des fichiers contenant le noyau au niveau du répertoire /boot

    make install -j3

LAB-02 : Conception et implémentation d'appel système

- **Deuxième partie : Ajout et utilisation d'un premier appel système**
- Vous allez modifier légèrement le noyau Linux de votre machine virtuelle pour ajouter un nouvel appel système.
- L'appel système (***sys_log710a2018as1***) que vous allez ajouter va simplement afficher un message dans le journal du système (system log).
- Cet appel système utilise la fonction noyau **printk** qui est l'équivalent de la fonction **printf()** de la librairie C standard. .

LAB-02 : Conception et implémentation d'appel système

- **Deuxième partie : Ajout et utilisation d'un premier appel système**
- **Déclaration de l'appel système**
  - Modifier la table des appels système (le fichier **syscall_32.tbl** pour 32 bit et **syscall_64.tbl** pour 64 bits) qui se trouve dans le répertoire **noyauSource/arch/x86/syscalls** pour définir le numéro du nouvel appel système (le nom de la fonction est **sys_log710a2018as1**).
  - Modifier le fichier (header) **syscalls.h** qui se trouve dans le répertoire **noyauSource/include/linux** en ajoutant la signature de la fonction de l'appel système.

```
asmlinkage long sys_log710a2018as1(void);
```

LAB-02 : Deuxième partie

- **Création et compilation du module**
  - Saisir l'implémentation de votre appel système sys_log710a2018as1 dans un nouveau fichier **noyauSource/partie2/syscall1_log710.c**.

```
#include <linux/kernel.h>
#include <linux/syscalls.h >
asmlinkage long sys_log710a2018as1(void) {
printk(KERN_ALERT "LOG710 AUTOMNE 2018 Appel Systeme 01!\n");
return 0; }
```

  - Créer un fichier **Makefile** spécifique à votre appel système.

```
obj-y := syscall1_log710.o
```

LAB-02 : Deuxième partie

- **Création et compilation du module**
  - Pour ajouter à la liste d'objets du noyau à compiler:l'objet **syscall1_log710.o**

  Ajouter le chemin **/partie2** pour l'attribut **core-y** dans le fichier Makefile du noyau.
  - Pour gagner du temps, surtout s'il y a une erreur dans le module, il est préférable de procéder la compilation du module de l'appel système avant la compilation du noyau

    make M=/partie2
- **recompiler et réinstaller le noyau (sans les modules)**

LAB-02 : Deuxième partie

- Ecrire et compiler un programme utilisateur pour tester l'appel système (voir l'énoncé).

- Consulter le fichier journal du système

  cat /var/log/ syslog

LAB-02 : Conception et implémentation d'appel système

- **Troisième partie : Appel système retournant des infos sur le processus courant**
- implémenter un nouvel appel système qui fournit des informations sur le processus courant (celui qui fait cet appel système).
  - Cet appel système ***sys_log710a2018as2*** prend comme paramètre un pointeur sur une structure de données ***procdata*** dans l'espace utilisateur.
  - L'appel système va mettre dans cette structure de données les informations sur le processus courant.
  - L'appel système retourne zéro si l'appel est avec succès ou une indication d'erreur (-EFAULT) sinon.

LAB-02 : Troisième partie

- Utiliser la structure **task_struct** retournée par la macro **current.** Cette structure contient la plus part des informations demandées.
- Utiliser la fonction **copy_to_user** pour transférer ces informations à l'espace utilisateur (***procdata***) désigné par le pointeur envoyé en parameter.
- Concevoir et implémenter un programme de test dans l'espace utilisateur qui invoque le nouvel appel système.
- Ce programme de test doit imprimer toutes les informations retournées par l'appel système. En particulier, tester le cas où le pointeur fournit à l'appel système est **NULL** ou **invalide**.

Questions?