Architecture d'un système d'exploitation

(ASE 2022-2023)

TD1

L'ordonnanceur du noyau Linux

hugo.taboada@cea.fr
marc.perache@cea.fr

Click here or scan to download TD'files from pcloud link.

Le but de ce TD est de démystifier le code du noyau ainsi que de comprendre de quelle façon est codé un ordonnanceur. Pour cela, une machine virtuelle est mise à disposition afin de faciliter la compilation du noyau (*$make bindeb-pkg* puis *#dpkg -i ../linux-image-4.9.30_4.9.30-X_amd64.deb* avec X le noyau choisi). Ensuite il suffit de redémarrer la machine virtuelle pour utiliser le nouveau noyau. Notons que nous pouvons nous connecter sur la machine virtuelle avec *ssh -p3022 ensiie@127.0.0.1* et le mot de passe est : ensiie. Le mot de passe root est : root.

I Fonctionnement de l'ordonnanceur

D'abord, nous vous proposons d'étudier le fonctionnement d'un ordonnanceur. Pour cela, nous allons parcourir le code source dans *kernel/sched/*.

Q.1: Quand avons-nous besoin de l'ordonnanceur ?

Q.2: Dans le code source, trouvez le point d'entrée de sched_yield(). À quoi correspond cette fonction ?

La fonction **printk** est une fonction permettant d'afficher des messages depuis le code dans le kernel. Nous pouvons visualiser les messages avec la commande **dmesg**.

Q.3: Insérez un **printk("current->se.exec_start=%llu\n",current->se.exec_start)** avant et après l'appel à la fonction **schedule()**. Écrivez un code utilisateur avec deux appels à **sched_yield()**. Regardez les messages avec dmesg. Que remarquez-vous ?

Q.4: Donnez l'enchaînement des fonctions depuis ce point d'entrée.

Q.5: Quelle est la fonction principale de l'ordonnanceur ?

Q.6: Pourquoi la préemption est-elle désactivé dans l'ordonnanceur ?

Q.7: Pourquoi le processus *next* libère-t-il le verrou de la run queue pris par le processus précédent ?

Q.8: Dans quelle fonction est choisi le prochain processus à exécuter ?

Q.9: Décrivez le fonctionnement de l'ordonnanceur. (explication + schéma)

Questions bonus :

Q.10: Trouvez le code du changement de contexte en assembleur et l'expliquer.

Q.11: Pourquoi cet ordonnanceur est bien adapté au temps partagé (plus de processus que de coeurs). Est-il adapté au HPC ?

II Abstractions dans l'ordonnanceur

Maintenant, nous allons étudier la façon dont les différents algorithmes d'ordonnancement sont reliés à l'ordonnanceur.

Q.12: Nous avons vu dans la question **Q.8**, la fonction permettant de sélectionner le prochain processus à exécuter. Quel mécanisme de programmation est utilisé pour relier les algorithmes d'ordonnancement et l'ordonnanceur ?

Q.13: Citez différents algorithmes utilisés dans le noyau linux.**Bonus : Expliquez-les.**

III Completely Fair Scheduling

Actuellement, l'algorithme utilisé par défaut dans le noyau linux est Completely Fair Scheduling (CFS).

Q.14: Dans quel fichier est implémenté cet ordonnanceur ?

Q.15: Sur quelle structure de donnée est basée cet algorithme ?

Q.16: Comment fonctionne-t-il ?

Ce TD est noté. Un rapport répondant aux questions est à envoyer avant le 09/10/2022 à 23h59 sur le lien pcloud suivant :
https://e.pcloud.com/#page=puplink&code=jM1Z7Q04SNEA2Hf0QMQlY91984GB6Bby

Pour toutes questions n'hésitez pas à m'envoyer un mail à l'adresse suivante :
hugo.taboada@cea.fr