210 Threads

INF3173
Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

Hiver 2021

Thread système

Fil d'exécution indépendant d'un processus

- On parle parfois aussi de « processus léger »

Pas d'isolation des threads

- Même programme
- Même contexte
- Mêmes ressources

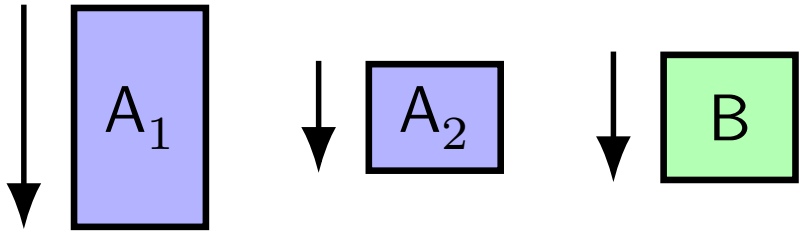
Mais exécution concurrente

- Permet des modèles de programmation intéressants
- Mais souvent difficiles...

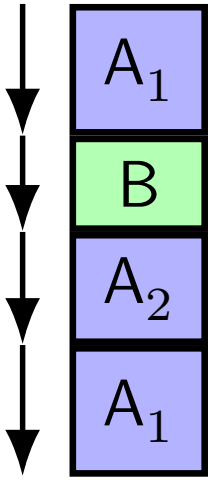
Exemple

- Un processus A avec 2 threads A_1 et A_2
- Un processus B monothread

Chaque thread progresse indépendamment



L'illusion de parallélisme est maintenue



Thread vs. Processus

Pour les threads d'un même processus

Propre à chacun

- Des registres (dont le CO et PP)
- Une pile d'exécution (pointée par PP)
- Priorité d'exécution

Partagé (en général)

- Programme en cours d'exécution (et bibliothèques)
- Sections mémoires (dont le tas)
- Fichiers ouverts

Avantages et inconvénients des threads

Avantages

- Moins cher à créer (un peu)
- Changement de contexte moins cher (un peu)
- Partage de données plus facile

Inconvénients

Synchronisation (très) difficile (on y reviendra)

- Un bogue dans un thread corrompt les autres
- Un thread compromis, compromet les autres

$\rightarrow$ Les navigateurs web modernes sont passé d'un thread par onglet à un processus par onglet

Programmation multithreads

Avant tout un **modèle de programmation**

Exposé par

- Des langages de haut niveau
- Des bibliothèques

Pour résoudre des problèmes variés

- Interface graphique réactive
- Traitements réseau asynchrones
- Calcul haute performance

Défis spécifiques

- Voir INF5171 Programmation concurrente et parallèle

Modèles d'implémentation multithread

Programmation multithread $\neq$ threads système

Plusieurs modèles d'implémentation multithread existent

Thread système (1:1)

- Le langage expose les threads système
- Le programmeur les manipule directement

Thread utilisateur (N:1)

- Les threads sont 100% gérés par le processus
- Le SE ne voit rien

Modèle hybride (M:N)

- C'est compliqué...

Thread utilisateur (N:1)

Géré 100% par le processus

- Offert souvent par des VM de langages
- Avec des astuces de programmation
- On parle aussi de *green thread*

$\rightarrow$ Juste un gros processus monothread compliqué

Avantages

- Portable entre différents SE
- Plus efficace dans certaines conditions
  - Pas de changement de contexte noyau

Inconvénients

- Changement de contextes utilisateur complexe à programmer
- Entrées-sorties bloquantes bloquent tout le processus
- Profite mal de la gestion optimisée des threads systèmes
- Profite mal des architectures multi-cœurs

Thread Posix (ou pthreads)

- API portable entre systèmes Unix
- Pour la programmation multithread système (1:1)
- Profite des threads système de chaque système d'exploitation
- voir le man `pthreads(7)` pour le point d'entrée
- On y reviendra...

Thread Linux (depuis v2.6, 2003)

- « **task** » (tâche) : seule abstraction de base
- Un processus monothread est juste une *task*
- Un processus multithread est un ensemble de *task*
  - Appartiennent à un même « *thread group* »
  - Un thread principal (*thread group leader*) représente le processus en entier
  - Le PID du processus est l'ID du thread principal

Voir et utiliser les threads Linux

Pour les voir

- `ps(1)`: `ps -Lf -C mysqld`
- `proc(5)`:
  - `/proc/` a une entrée **invisible** par thread
  - `/proc/PID/task/` pour les threads d'un même processus (« *thread group* »)

```
$ ls -l /proc/$(pgrep mysqld)/task
```

Pour programmer avec

Attention : Les threads Linux ne sont pas portables aux autres Unices

- Appels système spécifiques bas niveaux: `clone(2)`, `gettid(2)`, `tgkill(2)`...
- Pour du « vrai code », utiliser les `pthreads(7)`

Confusion terminologique

En fonction du contexte (et de l'auteur de la documentation)

- « processus (*process*) » peut désigner un processus ou un thread
- On trouve aussi « tâche (*task*) » pour compliquer plus

Autres regroupements de processus

- Groupe de processus: généralement utilisé pour les conduites shell (*pipelines*)
- Session: généralement utilisé pour grouper les connexions indépendantes d'utilisateurs