ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020

TD 10 – Concurrence, incohérence et synchronisation

Julien Sopena

décembre 2022

Le but de cette dixième semaine est de mettre évidence que la commutation entre processus peut être à l'origine d'incohérence sur des ressources partagées. Nous y verrons aussi comment répondre à ce problème en utilisant un mécanisme de verrouillage.

Exercice 1 : Exploiter au mieux ses ressources

Dans l'ensemble de cet exercice, sauf indication contraire, on supposera que les commutations (changements du processus élu) ont lieu uniquement sur les instructions `sleep` et `iss_synchro`.

Question 1

Nous allons étudier ici le code d'un script qui utilise la commande commande `shift n` qui permet de supprimer les `n` (1 par défaut) premiers paramètres et décaler ceux qui sont restants. Ainsi, le paramètre précédemment accessible avec `$4` l'est avec `$1` après exécution de la commande `shift 3`. Notons que `$0` ne change pas, car cela permet d'accéder à la commande lancée et non à un paramètre.

Que fait le script `add.sh` suivant ?

add.sh

```
#! /bin/bash

function usage {
  echo "Usage : $0 <fichier> <titre1> [liste titres]"
  exit 1
}

[ $# -lt 2 ] && echo "Il faut au moins deux paramètres." && usage

[ -d "$1" ] && echo "Le premier paramètre doit être un fichier." && usage
l=$1
shift

for f in $@ ; do
  if [ ! -f "$f.txt" ] ; then
    sleep 1
    echo "Fiche sur : $(echo $f | tr _ ' ')" > $f.txt
    echo "$f" >> $l
  fi
done
```

Question 2

Quel est le résultat de l'exécution des commandes suivantes ?

```
moi@pc /home/moi $ ./add.sh bibli.lst "Le_lièvre_de_Vatanen" "Le_loup_bleu" "Cyrano"
moi@pc /home/moi $ ./add.sh bibli.lst "Le_loup_bleu" "Comprendre_le_pouvoir"
```

Question 3

On exécute maintenant la commande suivante, quels sont les nouveaux contenus du répertoire et du fichier `bibli.lst` ? Vous appuierez votre réponse avec un chronogramme.

```
moi@pc /home/moi $ ./add.sh bibli.lst "Fondation" & ./add.sh bibli.lst "Fondation"
```

Question 4

Quel serait le résultat de la question précédente si l'on supprimait la commande `sleep,`, ainsi que l'hypothèse sur la commutation faite au début de l'exercice ?

Question 5

Pour résoudre ce problème, nous allons avoir besoin d'un mécanisme de synchronisation. Il en existe de toute sorte et dans tous les langages. Dans la suite de ce TD, comme en TP, vous utiliserez la commande `iss_synchro` qui fonctionne de la manière suivante :

- `iss_synchro create <nom_verrou>` : Création du verrou ;
- `iss_synchro destroy <nom_verrou>` : Destruction du verrou ;
- `iss_synchro lock <nom_verrou>` : Prendre le verrou, en attendant au besoin qu'il soit libéré en passant à l'état bloqué ;
- `iss_synchro unlock <nom_verrou>` : Libérer le verrou et réveiller les processus qui l'attendent.

En utilisant ce mécanisme de synchronisation, implémentez une nouvelle version du script (`add_sync.sh`) qui puisse s'exécuter correctement en parallèle (vous conserverez le `sleep` à son emplacement initial). Vous pouvez considérer qu'un verrou `myLock` a déjà été créé.

Question 6

Dessinez un chronogramme, correspondant à l'exécution de la commande suivante :

```
moi@pc /home/moi $ ./add_sync.sh bibli.lst "Fondation" & ./add_sync.sh bibli.lst "Fondation"
```

Question 7

Quel est le résultat de la commande suivante :

```
moi@pc /home/moi $ ./add_sync.sh bibli.lst "Fondation" & ./add.sh bibli.lst "Fondation"
```

Question 8

Implémentez maintenant un script `del.sh` qui effectue l'inverse de la fonction `add.sh`, *i.e.*, qui efface les fichiers correspondant aux ouvrages passés en paramètres et les supprime dans la liste passée en premier paramètre. Pour implémenter ce dernier point, vous passerez par la création d'un fichier temporaire qui, une fois édité, remplacera la liste originale. Voici un exemple d'utilisation de ce script :

```
moi@pc /home/moi $ ./del.sh bibli.lst "Mangez-le_si_vous_voulez" "Dans_les_forêts_de_sibérie"
```

Question 9

On veut trier l'ensemble des ouvrages de la liste avec le script `sort_bib.sh` suivant. Est-ce que ce script est compatible avec l'exécution de votre script `del.sh` ? Si oui pourquoi ? Sinon quelles sont les incohérences possibles ? Vous donnerez un scénario conduisant à ces incohérences.

sort_bib.sh

```
#! /bin/bash

function usage {
  echo "Usage : $0 <fichier>"
  exit 1
}

[ $# -ne 1 ] && echo "Il faut un paramètre." && usage

[ ! -f "$1" ] && echo "Le premier paramètre est un fichier." && usage
list=$1
tmp=$list.tmp

# Notons qu'il n'y a pas de doublon dans la liste des ouvrages
sort $list > $tmp
mv $tmp $list
```

Question 10

Est-ce que l'utilisation de commandes assurant à la fois la transformation et la substitution, comme avec l'option `-i/--in-place` de la commande `sed` dans ces deux scripts, est une solution au problème mis-en évidence à la question précédente.

Question 11

Proposez une correction de ces scripts qui permette de rendre possible une exécution parallèle des scripts `del.sh` et `sort_bib.sh`. Vous chercherez à conserver à maximum le parallélisme.