UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences

INFO-F-201 — Systèmes d'exploitation 1
J. Goossens
Année académique 2013–2014

Examen de Janvier 2014

Remarques préliminaires

— On vous demande de répondre à chaque question sur des feuilles séparées (les correcteurs sont différents).
— N'oubliez pas d'inscrire vos nom et prénom sur chacune des feuilles.
— Vous disposez de deux heures trente.
— Vous ne pouvez pas utiliser de notes (à l'exception du formulaire fourni avec l'énoncé).
— Vous devez fournir au moins une feuille par question (même si vous ni répondez pas).

Question 1 — Question Bash (8 points)

Nous vous demandons d'écrire un convertisseur de fichier avec parcours récursif dans une arborescence de fichiers. Vous devrez à partir d'un dossier de départ, passé en argument à votre programme($1), parcourir tous ses fichiers et ses sous-dossiers (et ainsi de suite récursivement) pour convertir les fichiers avec l'extension `.zoup` en fichier au format `.zap`.

Dans un fichier de type `.zoup`, tout est stocké ligne par ligne au format texte. Chaque ligne comporte six colonnes d'informations séparées par le caractère `#`. Vous les convertirez en fichiers `.zap` qui stockent les informations au format texte ligne par ligne, mais dont le séparateur pour chacune des six colonnes sur chaque ligne est cette fois `**`. Ci-dessous vous retrouverez un exemple de ligne dans le fichier au format `.zoup` et son équivalent dans un ficher au format `.zap`

```
... fichier .zoup ...
Chantal#756#1080#Rue des bouchers#12345678#banane
...

... fichier .zap ...
Chantal**756**1080**Rue des bouchers**12345678**banane
...
```

Vous ne pourrez pas utiliser `find`, `ls -r` (mais `ls` sans option est accordé) et `grep`. Vous considérerez que les fichiers portant les extensions présentées dans cette question sont corrects (il n'y aura donc aucune ligne mal formée).

Question 2 — Question 2 : La factorisation (12 points)

On vous demande un programme `C` qui a pour but le calcul de la factorielle d'un nombre `n`. Le programme recevra de sa ligne de commande deux informations : le nombre `n` ainsi que le nombre de threads `t` à utiliser pour le calcul.

Chaque thread calculera une partie de la factorielle. Autrement dit, l'intervalle `[1, n]` est découpé en `x` sous-intervalles (définies par leurs bornes inférieures et supérieures), chaque thread calcule le produit des entiers appartenant à un sous-intervalle. En toute évidence, le nombre `x` est déterminé en fonction du nombre de threads.

Si à la compilation, l'exécutable est appelé ‘‘`fact`’’, nous aurons par exemple :

```
./fact 5 2
```

Qui aura pour effet la création de 2 threads calculant la factorielle de 5. Les deux threads partagent une variable globale `res` initialement égale à `1` : le premier thread calculera le produit des entiers dans l'intervalle `[1, 2]` (et mettra à jour la variable `res` avec le produit `2`), tandis que le deuxième calculera le produit des nombres appartenant au sous-intervalle `[3, 5]` (et mettra à jour la variable commune avec le produit `60`).

Le programme affichera :

```
5!=120
```

Quels sont les avantages et inconvénients de l'utilisation des threads pour implémenter cette solution quant au niveau de la simplicité qu'au niveau de l'efficacité ? (pour répondre à cette question, pensez aux difficultés rencontrées si l'on n'avait pas pu utiliser les threads). Justifiez soigneusement vos réponses.