Programmation Système

Juliusz Chroboczek

15 juin 2009

La durée de l'examen est de 2 heures 30. Les documents sont autorisés, le matériel électronique est interdit.

Tous les programmes demandés devront être écrits en C, et les solutions dans d'autres langages de programmation ne seront pas acceptées. La notation prendra en compte la qualité de la gestion des situations exceptionnelles (« erreurs »).

Les parties sont indépendantes. Il est fortement conseillé de lire l'énoncé, qui est constitué de 2 pages.

1 Entrées/sorties fichiers

Dans cet exercice, l'utilisation de la bibliothèque `stdio` n'est pas autorisée, sauf pour présenter des messages d'erreur à l'utilisateur.

Écrivez un programme qui prend un nom de fichier en paramètre de ligne de commande et remplace ce fichier par un fichier obtenu à partir de ce dernier en remplaçant chaque suite de deux astérisques « ** » par un accent circonflexe « ^ ». (Le fichier d'entrée et le fichier de sortie portent le même nom, mais il n'est pas nécessaire de préserver l'identité du ficher transformé.)

2 Parcours d'arborescence

Dans cet exercice, l'utilisation des bibliothèques « `ftw` » et « `fts` » n'est pas autorisée.

Le *répertoire temporaire* est un répertoire où les programmes stockent leurs fichiers temporaires. Sa localisation est donnée par la variable d'environnement « `TMPDIR` » ; si celle-ci n'est pas définie, le répertoire temporaire est « `/tmp` ».

Normalement, les programmes suppriment les fichiers temporaires qu'ils créent ; cependant, il arrive qu'un programme termine de façon anormale (« se plante ») en laissant des fichiers temporaires.

Écrivez un programme qui parcourt l'arborescence enracinée au répertoire temporaire et supprime tous les fichiers qui n'ont pas servi depuis une semaine (i.e. dont le temps de dernier accès *(atime)* est inférieur à la date actuelle moins une semaine). Si, après suppression éventuelle de fichiers, un répertoire est vide, il devra être supprimé ; cependant, le répertoire temporaire lui-même ne devra pas être supprimé même s'il est vide. *(Indication : l'appel système `rmdir` échoue avec `errno` valant `ENOTEMPTY` lorsqu'il est appliqué à un répertoire non vide.)*

3 Processus

La commande « who » affiche une ligne pour chaque utilisateur connecté à la machine sur laquelle on l'exécute. Par exemple,

```
$ who
jch      :0           2009-06-15 09:33
jec      pts/0        2009-06-15 09:24
```

Écrivez un programme qui affiche le nombre d'utilisateurs connectés à la machine sur laquelle il est exécuté. Votre programme ne devra afficher que le nombre d'utilisateurs (un entier). Il ne devra exécuter aucun programme autre que « who » (en particulier, l'utilisation de la fonction « system », qui exécute un shell, n'est pas autorisée).