Granularité de l'exclusion mutuelle

On souhaite faire des statistiques sur les catégories des caractères contenus dans plusieurs fichiers. L'analyse consiste à maintenir quatre compteurs distincts, pour quatre catégories : `alnum`, `punct`, `space` et `other`. Les fonctions `isalnum()`, `ispunct()` et `isspace()` (déclarées dans `ctype.h`) permettent de reconnaître les trois premières catégories ; la dernière contient tous les autres caractères. On admet que les quatre compteurs sont des variables globales.

On veut écrire un programme (`ctypes.c`) qui prend en argument les noms des fichiers à traiter et lance un thread distinct pour chaque fichier. Chaque thread lit les caractères de son fichier un par un (avec `fgetc` par exemple), et pour chaque caractère incrémente le compteur de la catégorie correspondante. Lorsque tous les threads ont terminé leur analyse, le programme doit afficher les valeurs des quatre compteurs.

Puisque les différents threads accèdent concurremment aux compteurs, il faut assurer l'exclusion mutuelle lors des incrémentations. On hésite entre trois solutions :

1. utiliser un unique mutex pour protéger les accès aux quatre compteurs globaux ;
2. utiliser quatre mutex distincts pour protéger individuellement chaque compteur global ;
3. dans chaque thread, maintenir quatre compteurs locaux et ajouter une seule fois à la fin du traitement les valeurs aux compteurs globaux.

Programmez ces trois versions en utilisant la compilation conditionnelle et mesurez les performances de chaque version, par exemple en traitant tous les fichiers `/usr/include/*.h`. Que concluez-vous ?