Exercice 2

Difficulté : 77 points-virgules

On souhaite obtenir des informations sur les fichiers d'une arborescence : numéro d'inode, taille et nombre de caractères alphabétiques. Pour cela, on vous demande de réaliser un programme avec la syntaxe suivante :

```
infos répertoire
```

Le programme doit afficher, récursivement pour chaque fichier de l'arborescence, son numéro d'inode, sa taille en octets, le nombre de lignes et le nombre de lettres (on se limitera aux caractères non accentués). La sortie doit être triée par numéros d'inode. Par exemple :

```
> ./infos /tmp/test
12345 12 1 6 /tmp/test/x                # inode 12345, 12 octets, 1 ligne, 6 lettres
12356 789 25 521 /tmp/test/d/p          # exploration récursive des répertoires
12367 54 3 28 /tmp/test/y
12378 674 27 21 /tmp/test/d/d1/d2/z     # récursion à une profondeur quelconque
```

Pour rédiger votre programme, il est impératif de respecter les contraintes suivantes :

- vous ne devez utiliser que les primitives système (ou assimilées comme telles) ; vous pouvez toutefois utiliser les fonctions de bibliothèque pour les affichages ou les manipulations de chaînes de caractères ou de mémoire ;
- il est suggéré d'utiliser les fonctions de bibliothèque `isalpha` pour tester si un caractère est une lettre et `qsort` pour effectuer le tri demandé ;
- pour des raisons d'efficacité, vous ne ferez pas d'appels redondants à des fonctions lentes (primitives système ou autres) ;
- toujours pour des raisons d'efficacité, vous effectuerez tant que psosible des lectures ou d'écriture par quantité de `MAXBUF` octets que vous définirez à 4096 octets ;
- pour des raisons de simplicité, vous limiterez la taille des chemins à la constante `CHEMIN_MAX` que vous définirez à 128 octets : un chemin plus long doit être considéré comme une erreur ;
- vous vérifierez soigneusement les débordements de tableau (vous pouvez notamment utiliser la fonction de bibliothèque `snprintf` pour contrôler la taille de chaînes complexes) ;
- votre programme doit retourner un code de retour nul (`exit(0)`) si tout s'est déroulé sans erreur ou un code de retour non nul (`exit(1)`) si une erreur a été rencontrée ;
- si votre programme est appliqué avec un nombre d'arguments incorrect, il doit afficher un message de la forme : "`usage : infos repertoire`".
- vous apporterez un soin particulier à la mise en forme de façon à rendre un code lisible et commenté à bon escient. Référez-vous au document « Conseils pour réussir vos TP et projets » mis à votre disposition sur Moodle et, si besoin, utilisez l'utilitaire `clang-format` avec la configuration donnée dans ce document ;
- votre programme doit compiler avec les options `-Wall -Wextra -Werror -pedantic` sur `gcc` version 9.4 minimum (la version disponible sur la machine `turing.u-strasbg.fr`. Alternativement, vous pouvez utiliser l'image Docker `pdagog/refc` (version de `gcc` 13.2) Les programmes qui ne compilent pas au moins sur `turing` avec ces spécifications **ne seront pas examinés**.

Un script de test est mis à votre disposition sur Moodle. Celui-ci exécute votre programme sur des jeux de tests qui serviront de base à l'évaluation de votre rendu. La commande suivante permet de lancer les tests : `sh test2.sh`.

Vous devrez rendre sur Moodle un *unique* fichier nommé `infos.c`.

Cet exercice est **individuel**. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.