Exercice 2

Difficulté : 76 points-virgules

On souhaite identifier tous les liens physiques vers les mêmes fichiers dans une arborescence. Pour cela, on vous demande de réaliser un programme ayant la syntaxe suivante :

```
liens répertoire
```

Dans l'exemple suivant, le programme recherche récursivement dans toute l'arborescence dont la racine est le répertoire `/tmp/test` les différents chemins correspondant au même fichier, c'est-à-dire au même inode :

```
> ./liens /tmp/test
/tmp/test/d1/d11/a /tmp/test/d1/d12/b /tmp/test/d2/c      # un fichier avec 3 noms
/tmp/test/d2/d21/d /tmp/test/e                            # un fichier avec 2 noms
```

Ici, les trois chemins sur la première ligne (`/tmp/test/d1/d11/a` et suivants) correspondent au même fichier et les deux chemins sur la deuxième ligne (`/tmp/test/d2/d21/d` et suivant) correspondent au même fichier. Les chemins correspondant à un unique fichier ne doivent pas être affichés par votre programme.

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 ;
- pour des raisons d'efficacité, vous ne ferez pas d'appels redondants à des fonctions lentes (primitives système ou autres) ;
- pour des raisons de simplicité, vous limiterez la taille du chemin des fichiers créés à la constante `CHEMIN_MAX` que vous définirez à 512 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 : liens 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` 12.2) 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 `liens.c`.

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