Exercice 4

Difficulté : 86 points-virgules

On souhaite compter en parallèle le nombre d'occurrences d'un caractère donné dans les fichiers d'une arborescence. Pour ce faire, on demande de rédiger un programme avec la syntaxe suivante :

```
parcount répertoire caractère nproc
```

Par exemple :

```
> ./parcount /usr/include "#" 3
/usr/include/ctype.h 110 0              # 110 « # » dans ctype.h par le processus 0
/usr/include/netipx/ipx.h 32 1          # 32 « # » dans netipx/ipx.h par le processus 1
/usr/include/errno.h 11 1               # 11 « # » dans errno.h par le processus 1
/usr/include/stdio.h 214 0              # 214 « # » dans stdio.h par le processus 0
/usr/include/zlib.h 98 2                # 98 « # » dans zlib.h par le processus 2
...
```

Votre programme doit fonctionner suivant la méthode décrite ci-dessous.

1. Vous devez tout d'abord créer deux tubes : le premier (le tube d'*entrée*) servira pour fournir les chemins des fichiers sélectionnés et sera lu par tous les `nproc` processus chargés de compter les caractères, le deuxième (le tube de *sortie*) est partagé par tous ces processus qui devront y écrire le chemin du fichier, le numéro du processus (entre 0 et `nproc`-1) ainsi que le nombre d'occurrences trouvées.
2. Vous devez ensuite créer les `nproc` processus chargés de compter le nombre d'occurrences. Ces processus sont en compétition pour lire des chemins dans le tube d'entrée. Sitôt un chemin lu, le processus compte le nombre d'occurrences dans le fichier, puis il écrit dans le tube de sortie le chemin, le numéro du processus et le nombre d'occurrences, puis le processus se remet à lire dans le tube d'entrée. Lorsque la fin des données est détectée dans le tube d'entrée, le processus se termine.
3. L'étape suivante consiste à créer un processus supplémentaire pour lire dans le tube de sortie et afficher les informations suivant le format indiqué dans l'exemple ci-dessus. Vous indiquerez en commentaire pourquoi il est indispensable de créer ce processus supplémentaire pour afficher les résultats.
4. Une fois ces créations effectuées, le processus père explore l'arborescence indiquée et écrit les chemins trouvés dans le tube d'entrée ;
5. L'étape finale consiste à attendre la terminaison des `nproc`$+1$ processus et de vérifier s'ils se sont tous correctement terminés.

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 ; la génération de nombres pseudo-aléatoires ;
— 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 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 le message : `"usage : parcount répertoire caractère nproc"`.
— 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 test4.sh`.

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

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