Exercice 3

La semaine dernière, vous avez peut-être cherché à trouver un grand fichier dans votre arborescence. Cette semaine, vous allez rédiger un programme pour vous aider à trouver un tel fichier.

Ce programme doit avoir la syntaxe suivante :

```
tritaille répertoire
```

Il doit explorer l'arborescence indiquée par le répertoire pour afficher la liste des fichiers qui s'y trouvent (on ne s'intéresse qu'aux fichiers réguliers) triée selon la taille. Ainsi, par exemple, sur turing :

```
turing> ./tritaille /usr/include
        6 /usr/include/x86_64-linux-gnu/asm/setup.h
       19 /usr/include/x86_64-linux-gnu/sys/errno.h
       19 /usr/include/x86_64-linux-gnu/sys/fcntl.h
      ...
   391772 /usr/include/valgrind/valgrind.h
   410096 /usr/include/GL/glcorearb.h
   832367 /usr/include/GL/glext.h
```

Pour rédiger ce programme :

- vous n'imposerez pas de limite arbitraire quant à la taille de l'arborescence explorée : le seul facteur limitant est la quantité de mémoire disponible ;
- pour des raisons d'efficacité, vous ne ferez pas d'appels redondants aux primitives systèmes ;
- pour des raisons de simplicité, vous limiterez la taille des chemins à la constante `CHEMIN_MAX` que vous définirez à 512 octets : un chemin plus long doit être considéré comme une erreur ;
- vous vérifierez soigneusement qu'il n'y a pas de débordement de tableau (vous pouvez notamment utiliser la fonction de bibliothèque `snprintf` pour contrôler la taille de chaînes complexes) ;
- toujours pour des raisons de simplicité, vous ne traiterez que les fichiers réguliers et les répertoires dans une arborescence : tout autre type de fichier doit être ignoré ;
- pour trier un tableau, vous pouvez vous servir de la fonction de bibliothèque `qsort`.

On rappelle qu'il s'agit d'un travail **individuel**.