Exercice 1 - stat()

En utilisant l'appel système `stat()`, écrire un programme affichant le numéro d'inode, la taille, la date de modification et le type (fichier, répertoire, ou autre) d'un fichier passé en argument. Inutile de mettre en forme la date de modification : il suffit de l'afficher en l'état (en secondes depuis le premier janvier 1970). Quant au type, affichez juste "d", "f", ou "?" pour indiquer un répertoire, un fichier, ou autre chose. Pour connaître le type du fichier, utilisez les macros `S_ISDIR` et autres, sur le champ `st_mode` de la structure passée à `stat()`.

Exercice 2 - lstat()

Modifier le programme précédent pour indiquer si le fichier est un lien (par la lettre "l"); et donner la destination du lien. Pour cela, utiliser l'appel système `lstat()` (attention: `stat()` suit toujours les liens, et n'indiquera jamais si un fichier est un lien ou pas! Par ailleurs, si on demande à `stat()` de suivre un lien invalide, il renverra une erreur), et l'appel système `readlink()` pour connaître la destination du lien. Bien lire la manpage de `readlink()` ...

Exercice 3 - Ma mini-version de ls

En utilisant les fonctions de bibliothèque `opendir()` et `readdir()`, écrire un programme qui liste le contenu du répertoire courant, en affichant juste le nom de chaque fichier. N'oubliez pas le `closedir()` à la fin. Puis, afficher aussi les mêmes informations qu'à l'exercice précédent (réutiliser le code) - en ajoutant le nom de chaque fichier, bien sûr. Attention: il faut concaténer le nom du répertoire et le nom du fichier; nous conseillons l'utilisation de `snprintf()`.
Modifier le programme précédent afin que:

- s'il n'y a pas d'argument, il fonctionne en faisant l'affichage du répertoire "."
- si le premier argument est un fichier, il fonctionne comme le programme de l'exercice 2
- si le premier argument est un répertoire, il liste le contenu de ce répertoire
- s'il y a plusieurs arguments, il les traite les uns après les autres

Exercice 4 - Surveillance de fichier

En utilisant la date de modification, écrire un programme qui "surveille" un fichier fourni en argument jusqu'à ce que ce fichier soit modifié. Le programme devra consulter sans cesse la date de modification (avec `stat()`) jusqu'à ce qu'elle change; puis quitter. On peut utiliser les fonctions `sleep()` ou `usleep()` entre deux appels à `stat()` afin que le programme ne consomme pas tout le temps CPU. Rappel: pour modifier un fichier, on peut faire `touch le_fichier`, ou encore `echo >> le_fichier` par exemple. Tester le programme avec un répertoire. Question : quelles sont les opérations qui modifient un répertoire ?

Exercice 5 - Question bonus

Modifier le programme de l'exercice 3 afin qu'il puisse utiliser l'argument " `-R` " pour fonctionner de façon récursive, comme `ls`. Utiliser `getopt()` pour traiter les arguments de la ligne de commande.

Exercice 6 - Question Super bonus

Modifier le programme de l'exercice 3 afin qu'il puisse utiliser l'argument " `-R` " pour fonctionner de façon récursive, comme `ls`. Utiliser `ftw()` pour gerer le parcourt récursif.