Exercice 1 - Sur les traces de Richard Stallman ...

- Faire la question bonus du TD 1, en utilisant la fonction `getopt()` (voir exemple suivant) ou ce fichier

```c
#include <stdio.h>
#include <getopt.h>

int main(int argc, char * argv[]) {
    // -h -> help
    // -i input
    // -o [output]
    const char * optstring = ":hi:o::";
    int val;
    printf("optstring = %s \n", optstring);
    while (EOF != (val = getopt(argc,argv,optstring))) {
        printf("optind = %d \n", optind);
        printf("optarg = %s \n", optarg);
        switch (val) {
        case 'h':
            printf("help\n"); break;
        case 'o':
            printf("output %s\n",optarg); break;
        case 'i':
            printf("input %s\n",optarg); break;
        case ':':
            printf("arg missing for option %c\n",optopt); break;
        case '?':
            printf("unknown option %c\n",optopt); break;
        default:
            printf("default : unknown option %c\n",optopt); break;
        }
    }
}
```

- Reprendre le premier programme et modifier la fonction d'affichage afin qu'elle soit paramétrable. En fonction des paramètres, elle devra fonctionner comme " `ls` ", " `ls -l` ", ou " `ls -ln` ". Indications: pour convertir les UID et GID en noms, utiliser `getpwuid()` et `getgrgid()` ; pour convertir la date, utiliser `ctime()` ou une fonction similaire. **Attention:** attachez un soin tout particulier à écrire un code propre et compact pour afficher le type de fichier et les droits d'accès au fichier.
- Combiner les programmes des deux questions précédentes afin d'obtenir un " `ls` " comprenant les options `-R -n -l` (utilisez toujours `getopt()` pour l'analyse de la ligne de commande). Pour l'instant, il n'est pas nécessaire de trier les répertoires ni d'afficher leur taille totale, comme le fait le "vrai" `ls` . Il n'est pas nécessaire non plus de gérer la récursivité comme `ls` , qui liste le contenu du répertoire puis descend dans les sous-répertoire. Vous pouvez descendre dans les répertoires au fur et à mesure que vous les rencontrez.
- Modifier le programme pour qu'il utilise `scandir()` à la place d' `opendir()` / `readdir()` .
- Trier les répertoires par ordre alphabétique (comme `ls`); indice: il y a un paramètre `compar` dans `scandir()` qui permet de le faire très, très facilement.
- Écrire une fonction `isDirectory(const struct dirent*)` qui indique si la structure passée est ou non un répertoire.
- Utiliser cette fonction pour que le `ls` que vous avez écrit, lorsqu'il marche en mode récursif, liste d'abord le contenu du répertoire, puis descende dans les sous-répertoires.
- Question bonus: bravo, vous avez un `ls` qui marche. Si vous êtes arrivés jusque-là, implantez aussi les options `-a -d -i -s` . Pour comparer votre `ls` avec le `ls` "normal", utilisez ce dernier sans alias et avec l'option `-1` (moins un), par exemple " `/bin/ls -1` ".