TP à rendre 1

1 Énoncé :

Dans cet exercice vous devez écrire le programme `dirinfo` qui prend pour seul argument le chemin d'un dossier et affiche récursivement des informations sur tous les fichiers et sous-dossiers qu'il contient :

```
./dirinfo <chemin d'un dossier>
```

2 Consignes :

Votre programme doit parcourir récursivement tous les fichiers et sous-dossiers du dossier indiqué en argument. Pour chacun de ces dossiers, il doit lister les fichiers en indiquant leur nom, leur taille (en octets) et le format du fichier.

Le format du fichier est obtenu en lisant les premiers octets de chaque fichier et en les comparant à une liste de « nombre magique » connue [1]. Les seuls formats que votre programme doit comprendre vous sont donnés ci-dessous dans le Tableau 1. Par exemple, si les deux premiers octets d'un fichier ont respectivement pour valeur hexadécimale `50` et `4B`, alors il s'agit d'un fichier d'archive au format ZIP. Pour les fichiers étant des scripts avec *shebang* [2], votre programme devra alors afficher la première ligne du fichier (c'est-à-dire tous les caractères jusqu'au premier saut de ligne rencontré).

Nombres magiques (en hexadécimal)	Signification	Texte a afficher
`50 4B`	Archive zip	ZIP
`37 7A BC AF 27 1C`	Archive 7z	7z
`25 50 44 46 2D`	PDF	PDF
`49 44 33`	MP3	MP3
`49 49 2A 00`	Image TIFF	TIFF (little endian)
`4D 4D 00 2A`	Image TIFF	TIFF (big endian)
`23 21`	Script avec *shebang*	*La première ligne du fichier (#!...)*

Tableau 1 – Liste des nombres magiques à intégrer dans `dirinfo`

Pour vous aider à vérifier le bon comportement de votre programme, vous pouvez afficher les premiers octets d'un fichier grâce à la commande `xxd`. Exemple : `xxd -g 1 -len 150 <mon fichier>`

Si votre programme rencontre un dossier vide, votre programme n'a rien à faire pour celui-ci (c'est-à-dire, il l'ignore et continue son exécution).

Si votre programme rencontre un dossier ou un fichier inaccessible, non lisible, ou spécial (par exemple, fichier de type lien, block, etc), il doit afficher un message d'erreur et s'arrêter. D'une façon générale, si votre programme rencontre un cas non géré, alors il doit s'arrêter immédiatement après avoir affiché à l'utilisateur un message pertinent.

Enfin, comme pour tout bon programme, vous devez également vérifier l'argument qui lui est passé et afficher un message d'erreur adapté en cas de problème, voir les exemples en Section 3. Ces messages d'erreurs sont à afficher sur la sortie d'erreur, et le code de sortie de votre programme doit être 1 (0 en cas d'exécution sans erreur).

Vous apporterez également un soin particulier à la mise en forme de votre code de façon à rendre un code lisible et commenté à bon escient. Référez-vous au document « Conseils de programmation pour réussir vos projets et TP » (mis à votre disposition sur Moodle) et si besoin utilisez l'utilitaire 'indent' avec la configuration donnée en fin du document.

1. https://fr.wikipedia.org/wiki/Nombre_magique_(programmation)
2. https://fr.wikipedia.org/wiki/Shebang

Liste des primitives système qui vous aideront à démarrer : `readdir`, `opendir`, `lstat`, `read`

2.1 Programmation :

Vous écrirez le programme `dirinfo` en langage C en utilisant uniquement les primitives systèmes pour accéder aux fichiers et les fonctions de bibliothèques `opendir`, `readdir` et `closedir` pour les dossiers. Sinon, les seules fonctions de bibliothèque autorisées sont pour l'affichage et la comparaison de chaîne de caractères (par exemple : `fprintf`, `snprintf`, `strcmp`). Votre programme doit vérifier les valeurs de retour de toutes les primitives système utilisées. Pour toute opération sur des données, vous devez vous assurer que votre programme ne va jamais lire ou écrire en dehors de sa mémoire. Votre programme ne devra pas comporter d'allocation dynamique de mémoire.

Votre programme doit compiler sur la machine `turing.unistra.fr` avec la commande suivante (disponible dans le *Makefile* mis à disposition sur Moodle) : `cc -Wall -Wextra -Werror -Wvla`
Les programmes qui ne compilent pas avec cette commande **ne seront pas examinés**.

2.2 Tests :

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. N'hésitez pas à contacter votre enseignant si vous constatez un comportement anormal ou si vous souhaitez ajouter un test.

2.3 À rendre :

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

Moodle sait qui vous êtes, il est inutile d'appeler votre programme `Jean-Claude_Dusse_dirinfo.c`, ainsi il est interdit de rendre un fichier d'un autre nom ou une archive au format du jour. De plus, assurez-vous de rendre des fichiers utilisant l'encodage UTF-8 – il y aura de sévères pénalités sinon.

Ce TP à rendre est **individuel**. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.

3 Exemples de sortie attendue :

3.1 Utilisation correcte du programme :

```
$ ./dirinfo ./mon/dossier
./mon/dossier/fichier1.zip                          1400 octets    ZIP
./mon/dossier/sauvegardes/photos/fichier2.tiff      8060 octets    TIFF (little endian)
./mon/dossier/sauvegardes/photos/fichier3.tiff      4051 octets    TIFF (little endian)
./mon/dossier/sauvegardes/photos/fichier1.tiff      3600 octets    TIFF (big endian)
./mon/dossier/sauvegardes/archives/fichier2.zip     4200 octets    ZIP
./mon/dossier/sauvegardes/archives/fichier1.7z      1202 octets    7z
./mon/dossier/sauvegardes/archives/fichier4.zip        0 octets    Format inconnu
./mon/dossier/sauvegardes/archives/fichier5.abc      510 octets    Format inconnu
./mon/dossier/scripts/fichier3.toto                 7630 octets    Format inconnu
./mon/dossier/scripts/fichier4.sh                   1110 octets    #!/bin/sh
./mon/dossier/scripts/fichier1.php                  2860 octets    #!/usr/bin/env php
./mon/dossier/scripts/fichier2.sh                   1210 octets    #!/usr/bin/env bash
./mon/dossier/fichier2.pdf                          3180 octets    PDF
./mon/dossier/fichier3.mp3                          1400 octets    MP3
```

Code retour : 0

Remarque : il n'est pas demandé de fournir une sortie qui aligne les colonnes des différents fichiers listés comme sur cet exemple. Vous pouvez vous contenter d'un simple espace ou du caractère de tabulation pour séparer les informations.

3.2 Mauvaise utilisation du programme :

3.2.1 Oublie de l'argument :

```
$ ./dirinfo
Usage: ./dirinfo <chemin d'un dossier>
```

Code retour : 1

3.2.2 Plusieurs dossiers passés en argument :

```
$ ./dirinfo ./dossier1 ./dossier2
Usage: ./dirinfo <chemin d'un dossier>
```

Code retour : 1