Exercice 4

Le but de cet exercice est de rédiger un programme comparable à `tar`, dont les deux fonctions sont de créer un fichier *archive* contenant une arborescence et d'extraire l'arborescence d'un fichier *archive*. La fonction souhaitée est déterminée à partir des arguments :

- `arc c archive dir` : crée le fichier *archive* contenant l'arborescence indiquée par le répertoire *dir*
- `arc x archive` : extrait l'arborescence contenue dans le fichier *archive*.

Le format du fichier *archive* est imposé :

- il doit commencer par les 4 octets 0x63, 0x73, 0x6d et 0x69 : ils permettent aux applications (et notamment votre programme lorsqu'il fera une extraction) de reconnaître un fichier archive et de refuser l'extraction si ce n'est pas le cas ;
- il contient ensuite des entrées, pour chaque fichier ou répertoire, placées à la suite les unes des autres. Chaque entrée a le format suivant :
  - un octet valant 0x01 si c'est un fichier régulier, 0x02 si c'est un répertoire ou 0x03 si c'est un lien symbolique
  - 2 octets pour contenir les permissions (seuls les 9 bits de poids faible sont utilisés) de l'objet
  - 2 octets pour la longueur du chemin relatif de l'objet
  - puis le chemin relatif de l'objet (sans l'octet nul de fin de chaîne, sans objet puisqu'on a la longueur)
- les entrées correspondant à des fichiers réguliers ont des informations complémentaires venant immédiatement à la suite de l'entrée :
  - 4 octets pour la date de dernière modification
  - 8 octets pour la taille du fichier
  - puis les octets composant le fichier
- les entrées correspondant à des liens symboliques ont des informations complémentaires venant immédiatement à la suite de l'entrée :
  - 2 octets pour la taille du lien (chemin vers la cible)
  - puis le chemin vers la cible du lien (sans l'octet nul de fin de chaîne)

  On notera que les permissions indiquées dans l'entrée ne sont pas utilisées pour les liens.

Le respect du format décrit ci-dessus est indispensable : vos programmes seront testés par rapport à une implémentation de référence et toute incohérence sera pénalisée. Le fichier `exemple-exo4.tar` disponible sur Moodle contient 4 exemples d'archives selon ce format. Vous pouvez explorer leur contenu sous Linux à l'aide de la commande `hd` (paquet `bsdmainutils` sur Ubuntu et Debian). Il est suggéré d'utiliser ces exemples pour réaliser l'extraction dans un premier temps, puis d'implémenter la création dans un deuxième temps.

Quelques éléments vous sont utiles :

- le fichier d'inclusion `inttypes.h` contient les types `uint8_t`, `uint16_t`, `uint32_t` et `uint64_t` utiles dans les programmes manipulant des formats binaires tels que celui qui vous est imposé ;
- pour des raisons d'efficacité, vous ne ferez pas d'appels redondants aux primitives systèmes, et vous réaliserez le transfert du contenu d'un fichier de ou vers l'archive avec des blocs de `TAILLE_BLOC` octets (constante que vous définirez à 4096 octets) ;
- pour des raisons de simplicité, vous limiterez la taille des chemins (y compris les cibles des liens symboliques) à la constante `CHEMIN_MAX` que vous définirez à 512 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) ;
- toujours pour des raisons de simplicité, vous ne traiterez que les fichiers réguliers, les répertoires et les liens symboliques dans une arborescence : tout autre type doit être considéré comme une erreur ;
- vous détecterez une erreur si on essaye de créer une archive contenant un nom absolu ;
- vous pouvez prendre l'hypothèse que les répertoires sont bien rangés dans l'archive : un répertoire doit être placé avant les objets qu'il contient. Bien sûr, vous vous assurerez que cette hypothèse est vérifiée lors de la création ;
- les fichiers doivent être restaurés avec les bonnes permissions et les bonnes dates de modification, et les répertoires avec les bonnes permissions.