Université de Genève

-

Sciences Informatiques

Systèmes d'Exploitation - TP 03

Noah Munz - Gregory Sedykh

Octobre 2022

Contents

TP 03

4 Backup de dossiers et fichiers

4.1 Utilisation

4.1.1 Compilation

La commande `make` permet de compiler tous les fichiers et rendra un exécutable appelé `ultra-cp`. D'autres options sont disponible dans le makefile.
Une petite documentation avec les fichiers et leur utilité est listée ci-dessous:

1. `main.c`
   Contient 3 fonctions:
   (a) `handleArgs` : Permet de gérer la totalité des arguments passés (les fichiers/dossiers (obligatoire) et les arguments optionnels tels que `-a` et `-f`)
   (b) `ultra_cp` : Copie le contenu en fonction des arguments
   (c) `main` : Gère les differentes possibilités de copier les fichiers/dossiers (seulement 2 fichiers, 1 fichier et un dossier, et les arguments optionnels)
2. `copy.c`
   Contient 4 fonctions:
   (a) `is_modified` : Compare les temps de la dernière modification, et leurs tailles. S'il y a une différence entre les deux, le fichier sera remplacé par le nouveau
   (b) `copy` : Copie le contenu d'un fichier à un autre (code de l'exemple du chapitre 7: I/0 du cours, avec quelques modifications)
   (c) `copy_ifneeded` : Vérifie si deux fichiers diffèrent et copie si c'est le cas
   (d) `ultra-cp-single` : Copie avec les fonctionnalités optionnelles demandées.
3. `files.c`
   Contient 11 fonctions:
   (a) Les fonctions `is...` : Vérifient si l'argument est quelque chose. Ex: isDir vérifie si l'argument est un directory
   (b) `computePerm` : Retrouve les permissions d'un fichier donné
   (c) `exists` : Dit si un fichier donné existe. Retourne un erreur sinon
   (d) `concat_path` : Concatène le path d'un fichier et de son directory parent
   (e) `absPath` : Retourne le path absolu d'un fichier
   (f) `getFileName` : Retourne le vrai nom d'un fichier à partir du path absolu
   (g) `listEntry` : Print les informations d'un fichier: type, permissions, taille, heure de modification et son path
   (h) `listEntryNoIn` : Idem, mais sans inode en argument
   (i) `list_dir` : Print les informations de chaque fichier dans un directory

4. `optprsr.c`
   Contient 3 fonctions:
   (a) `checkEnoughArgs` : Vérifie que suffisamment d'arguments obligatoires ont été entrées
   (b) `parseArgs` : Parse les arguments obligatoires. Ici, ce sont les fichiers et dossiers sources et la destination
   (c) `parseOptArgs` : Parse les arguments optionnels. Ici, ce sont `-a` et `-f`

5. `util.c`
   Contient 9 fonctions:
   (a) `tryalc` : Vérifie que malloc a été effecté sans problème
   (b) Les fonctions `hdl...` : Gèrent multiples erreurs avec les fichers (ouvrir, fermer etc.)
   (c) `stat_s` et `lstat_s` : Permettent d'obtenir les informations d'un fichier (comme stat et lstat dans bash)

4.1.2 Exécution

`ultra-cp` peut être executé avec:

```
./ultra-cp dossier_1 ..  dossier_n fichier_1 ..  fichier_n destination [-a] [-f]
```

- S'il n'y a que 2 fichiers entrés, alors le contenu du premier (source) sera copié dans le deuxième (destination).
- Si la destination n'existe pas, alors elle est créée (seulement dans le cas où on a que 1 seul fichier en entrée).
- Le programme vérifie et ne copie que les regular files, les links et les directories.
- Les droits ne pourront pas être les mêmes dans le cas ou les droits du dossier source n'est pas le meme que ceux du dossier destination, c'est-à-dire que le umask de la destination est plus restrictive que celle la source.
  Par exemple, on ne pourra pas copier dans un dossier ou il y a des fichiers systèmes, nécissitant un `sudo` pour faire des manipulations avec.
- Le fichier ou le dossier de destination ne sera remplacé que s'il y a eu une modification de temps ou de taille.
- `-a` : L'utilisateur veut modifier les permissions (777 par défaut).
- `-f` : L'utilisateur ne veut pas deréférencer les liens. Les liens sont donc copiés comme liens.