

Systèmes d'Exploitation - Examen

12X009 - TP03

Ultra-cp

Noah Munz (19-815-489)

Département d'Informatique
Université de Genève

Mardi 31 Janvier 2023

Code: [Lien GitHub du code](#)

Rapport: [Lien GitHub du rapport](#)



Table of Contents

- 1 Rappel: But du TP
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



Table of Contents

- 1 **Rappel: But du TP**
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



Rappel: But du TP

- L'objectif général du TP est de créer un programme qui permet de faire des backups d'architecture de fichiers facilement, en s'inspirant du programme `rsync`.
- Le programme doit être capable de faire des backups incrémentaux, c'est à dire qu'il doit être capable de détecter les fichiers qui ont été modifiés depuis la dernière sauvegarde, et de ne sauvegarder que ceux-ci.

Les principaux défis de ce TP sont:

- La gestion de fichiers et de répertoires ainsi que leurs architectures.
- La gestion et manipulations d'attributs d'inodes.



Table of Contents

- 1 Rappel: But du TP
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



TADs & leurs relations: Décomposition modulaire

4 modules (+ `main.c`) ont été créés pour la réalisation du TP:

- ➊ `main.c` : contient le code principal du programme & appelle les fonctions appropriées des autres modules.
- ➋ `copy` : contient 4 fonctions, sert à gérer la copie des fichiers et répertoires, de définir quand est-ce qu'un fichier est considéré comme "modifié"...
- ➌ `files` : contient 11 fonctions, servant à gérer les fichiers et leurs attributs.
- ➍ `optprsr` : contient 3 fonctions, servant à gérer les options passées en paramètres. (repris du tp d'avant)
- ➎ `util` : contient 9 fonctions, servant divers usages allant de la gestion d'erreurs à la gestion de chaînes de caractères, en passant par des wrappers qui incluent lesdites fonctions de gestions d'erreurs.



L'implémentation de structure n'a pas été nécessaire.



TADs & leurs relations: Décomposition fonctionnelle

(1) `main.c`

- ▶ `handleArgs` : Permet de gérer la totalité des arguments passés (les fichiers/dossiers (obligatoire) et les arguments optionnels tels que `-a` et `-f`) utilise le module `optprsr` pour parse/extraire les arguments obligatoire & optionnels et leurs fait correspondre le bon chant du bitfield `ST_...` .
- ▶ `main` : Gère les différentes possibilités de copier les fichiers/dossiers (seulement 2 fichiers, 1 fichier et un dossier, 1 (ou plus) dossier(s) vers 1 dossier et les arguments optionnels) i.e. appelle `handleArgs` récupérer le *state* (bitfield `ST_...`) pour ensuite déléguer aux fonctions du module `copy` en fonctions des différents



(2) copy

- ▶ `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
- ▶ `copy` : Copie le contenu d'un fichier à un autre (code de l'exemple du chapitre 7: I/O du cours, avec quelques modifications)
- ▶ `copy_ifneeded` : Vérifie si deux fichiers diffèrent et copie si c'est le cas
- ▶ `ultra-cp-single` : Si le fichier passé en argument a été modifié
⇒ copie à la destination passée en argument; en changeant les permissions du fichier copié si demandé en option.
- ▶ `ultra_cp` : Backup le dossier passé en argument à la destination passée en argument. Traverse récursivement tous les dossiers en créant les dossiers nécessaires au passage et appelle `ultra-cp-single` pour chaque fichier rencontré.



(3) files

- ▶ Les fonctions `is...` : Vérifient si l'argument est quelque chose. Ex: `isDir` vérifie si l'argument est un directory
- ▶ `computePerm` : Retrouve les permissions d'un fichier donné
- ▶ `exists` : Dit si un fichier donné existe. Retourne un erreur sinon
- ▶ `concat_path` : Concatène le path d'un fichier et de son directory parent
- ▶ `absPath` : Retourne le path absolu d'un fichier
- ▶ `getFileName` : Retourne le vrai nom d'un fichier à partir du path absolu
- ▶ `listEntry` : Print les informations d'un fichier: type, permissions, taille, heure de modification et son path
- ▶ `listEntryNoIn` : Idem, mais sans inode en argument
- ▶ `list_dir` : Print les informations de chaque fichier dans un directory



(4) optprsr

- ▶ `checkEnoughArgs` : Vérifie que suffisamment d'arguments obligatoires ont été entrées
- ▶ `parseArgs` : Parse les arguments obligatoires. Ici, ce sont les fichiers et dossiers sources et la destination
- ▶ `parseOptArgs` : Parse les arguments optionnels. Ici, ce sont `-a` et `-f`



(5) `util`

- ▶ `tryalc` : Vérifie que `malloc` a été effectué sans problème
- ▶ Les fonctions `hdl...` : Gèrent multiples erreurs avec les fichiers (ouvrir, fermer etc.)
- ▶ `stat_s` et `lstat_s` : Permettent d'obtenir les informations d'un fichier (comme `stat` et `lstat` dans `bash`)



Table of Contents

- 1 Rappel: But du TP
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



Tests réalisés pour valider le fonctionnement du TP

Les tests réalisés ont été les suivants:

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

Selon les utilisations suivantes: (commentaire copié du code)

1. Just 1 file/folder \Rightarrow print/lis contents
2. If only 2 files are given \Rightarrow create/replace the file
3. If multiple files & folder and “dest” exists \Rightarrow create/replace architecture in “dest”
4. If -a is passed, change the permissions
5. If -f is passed, links are copied as links (stored in optional state)



Tests réalisés pour valider le fonctionnement du TP

Soit le dossier `res/` contenant les fichiers suivants:

```
❏ WSL at ~ > BA3 > 12X009-OS-TPs > TP03
```

`res/test_src/:`

<code>-rw-r--r--</code>	5	Thu Jan 5 15:32:45 2023	<code>res/test_src/foo.txt</code>
<code>-rw-r--r--</code>	0	Thu Jan 5 15:32:45 2023	<code>res/test_src/bar.txt</code>

`res/ (rest):`

<code>drwxr-xr-x</code>	4096	Thu Jan 5 15:32:45 2023	<code>res/fool</code>
-------------------------	------	-------------------------	-----------------------

`res/fool/:`

<code>-rw-r--r--</code>	5	Thu Jan 5 15:32:45 2023	<code>res/fool/foo.txt</code>
<code>-rw-r--r--</code>	3794	Thu Jan 5 15:32:45 2023	<code>res/fool/listDir.c</code>
<code>-rw-r--r--</code>	0	Thu Jan 5 15:32:45 2023	<code>res/fool/bar.txt</code>
<code>-rw-r--r--</code>	15	Thu Jan 5 15:32:45 2023	<code>res/fool/test.bak.txt</code>
<code>drwxr-xr-x</code>	4096	Thu Jan 5 15:32:45 2023	<code>res/fool/test1</code>

`res/fool/test1/:`

<code>drwxr-xr-x</code>	4096	Thu Jan 5 15:32:45 2023	<code>res/fool/test1/test2</code>
-------------------------	------	-------------------------	-----------------------------------

`res/fool/test1/test2/:`

<code>-rw-r--r--</code>	6	Thu Jan 5 15:32:45 2023	<code>res/fool/test1/test2/test1.test2.txt</code>
-------------------------	---	-------------------------	---

`res/fool/test1/ (rest):`

`res/fool/ (rest):`

<code>-rw-r--r--</code>	11	Thu Jan 5 15:32:45 2023	<code>res/fool/test.txt</code>
-------------------------	----	-------------------------	--------------------------------

`res/ (rest):`



Tests réalisés pour valider le fonctionnement du TP

Les tests suivants ont été réalisés:

- ① `./ultra-cp res/fool tmp -f`
- ② `./ultra-cp res/fool/bar.txt tmp.txt`
- ③ `./ultra-cp res/test_src/foo.txt`
- ④ `./ultra-cp res/test_src/foo.txt foo`
- ⑤ `./ultra-cp res/test_src/bar.txt res/test_src/foo.txt
res/test_src res/fool -f`

La dernière commande donnant l'output suivant:



Tests réalisés pour valider le fonctionnement du TP

```
[ WSL at ~ > BA3 > 12X009-OS-TPs > TP03
[ $ ./ultra-cp res/test_src/bar.txt res/test_src/foo.txt res/test_src res/fool
Copying /home/noahl/BA3/12X009-OS-TPs/TP03/res/test_src/foo.txt to /home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/foo.txt.

Copying /home/noahl/BA3/12X009-OS-TPs/TP03/res/test_src/bar.txt to /home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/bar.txt.
```

--- Backed up files ---

/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/:

-rw-r--r--	5	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/foo.txt
-rw-r--r--	3794	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/listDir.c
-rw-r--r--	0	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/bar.txt
-rw-r--r--	15	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test.bak.txt
drwxr-xr-x	4096	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1

/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1/:

drwxr-xr-x	4096	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1/test2
------------	------	-------------------------	---

/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1/test2/:

-rw-r--r--	6	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1/test2/test1.test2.txt
------------	---	-------------------------	---

/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test1/ (rest):

/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/ (rest):

-rw-r--r--	11	Thu Jan 5 15:32:45 2023	/home/noahl/BA3/12X009-OS-TPs/TP03/res/fool/test.txt
------------	----	-------------------------	--



Table of Contents

- 1 Rappel: But du TP
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



- Dans quels cas les droits du fichier copié ne pourront pas être les mêmes que ceux du fichier source?
 - ▶ Les droits ne pourront pas être les mêmes dans le cas où les droits du dossier source n'est pas le même que ceux du dossier destination, c'est-à-dire que le umask de la destination est plus restrictif que celle la source. Si on a pas les droits sur un dossier (e.g. /sys, /bin, /etc/mount ...) on ne pourra pas y copier de fichiers et donc par extension pas modifier les droits.



Table of Contents

- 1 Rappel: But du TP
- 2 TADs & leurs relations
 - Décomposition modulaire
 - Structures de données utilisées
 - Décomposition fonctionnelle
- 3 Tests réalisés pour valider le fonctionnement du TP
- 4 Réponses aux questions (de l'énoncé du TP)
- 5 Réponses aux questions (générales)



Réponses aux questions (générales)

Code: [Lien GitHub du code](#)

Rapport: [Lien GitHub du rapport](#)

