M1 EIDD 2017
Projet Systèmes et Réseaux v2017.1.2

Enseignant : Nicolas K. Blanchard (Nicolas.K.Blanchard@gmail.com)

11 mars 2017

Résumé

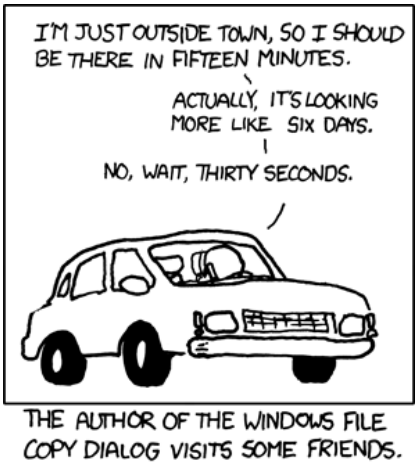
Le but de ce projet est d'implémenter un gestionnaire de fichiers ayant des fonctionnalités réseau, de type FileZilla. Ce projet représente la majeure partie de votre contrôle continu, représentant au moins 35% de la note finale. Une très bonne note sur le projet (ou l'implémentation de parties bonus) donnera aussi des points supplémentaires sur l'examen final.

1 Objectifs généraux

Le but du projet est de créer un gestionnaire de fichiers pouvant déplacer et renommer des fichiers entre plusieurs machines. Le système se compose d'un client et d'un serveur. Une fois le serveur lancé, le client doit pouvoir faire ce qui suit :

— Renommer, copier, coller, déplacer des fichiers sur sa machine et la machine hôte, que ce soit dans la même partitions, entre différentes partitions, sur différents disques ou entre machines.

— Gérer plusieurs opérations en même temps avec une file d'attente visible et avoir une estimation du temps restant (plus précise que l'estimation traditionnelle de Windows).



Ce qu'en pense Randall Monroe (XKCD).

— Être prévenu en cas d'opérations dangereuses ou impossibles, et des erreurs.

— Et enfin être le plus efficace possible.

Le serveur de son côté doit :

— Accepter un nombre quelconque de clients en simultané, et plusieurs opérations simultanées par client.

— Gérer les permissions pour que les opérations de ces clients ne se gênent pas mutuellement.

— Gérer les erreurs et exceptions pour ne pas planter dès qu'un client fait une opération interdite.

2 Fonctionnalités

Vous êtes libres de procéder comme bon vous semble sur le détail des fonctionnalités et l'interface utilisateur. Votre seul contrainte est de faire le logiciel le plus utile et le plus efficace possible. On fera au moins un minimum attention à l'UI. Attention, toutes les fonctionnalités usuelles UNIX ne sont pas autorisées (voir la partie Outils). Voilà une liste de fonctionnalités pour le client qui sont nécessaires pour avoir la moyenne. Du côté du serveur les contraintes sont celles expliquées précédemment.

2.1 Côté Client

Le client doit pouvoir choisir un lieu source – sur sa machine ou une autre – et si approprié un lieu cible, et faire les opérations suivantes :

Couper/Copier/Coller/Déplacer/Supprimer

Avec deux options, pour le faire par paquets ou directement en manipulant les inodes (attention au piège). Et peut-être la possibilité de gérer manuellement la taille des paquets.

Renommer

Attention aux caractères spéciaux.

Explorer l'arborescence

Gérer la file d'attente des opérations en cours

3 Fonctionnalités additionelles

Votre projet doit intégrer au moins trois de ces fonctionnalités, au choix. L'implémentation d'autres fonctionnalités supplémentaires augmentera naturellement votre note en proportion (chacune valant entre 1 et 5 points selon sa nature et la qualité de son implémentation). Vous pouvez aussi rajouter d'autres options de votre invention à votre guise en les expliquant dans votre rapport.

- La compatibilité avec le protocole FTP (faire un vrai FileZilla Lite) sera fortement récompensée et compte pour les trois fonctionnalités requises.
- L'implémentation d'une interface graphique sommaire (avec les librairies adaptées, votre prof n'étant pas sadique non plus – du moins pas avec vous) donnera pas mal de points et compte pour deux fonctionnalités additionnelles sur trois.
- Les outils de copie et renommage massifs.
- Un outil de gestion de file d'attente vous permettant de mettre en pause et de limiter la bande passante de certains transferts.
- La gestion des expressions régulières pour transmettre seulement certains fichiers.
- La connection sécurisée et chiffrée entre serveur et client.
- L'optimisation des transferts de nombreux petits fichiers.
- L'automatisation de transferts quotidiens.
- La synchronisation automatique de dossiers entre serveur et client.
- La gestion intelligente de plusieurs clients avec un partage équitable des ressources.
- La stabilité absolue de votre logiciel et des facilités de debugging (error log par exemple).
- La vérification de l'intégrité des fichiers envoyés (hash MD5 par exemple).

4 Outils

Vous avez le droit aux librairies graphiques de votre choix tant que vos dépendances sont bien gérées et justifiées, et que votre projet compile de manière indépendante. Vous n'avez pas le droit aux outils usuels de manipulation de fichiers UNIX, comme mv, rm, ls. Vous pouvez ouvrir et fermer des fichiers, et avoir les données des inodes, mais si un outil vous permet de faire une fonctionnalité en une ligne vous n'y avez pas droit en gros – vous pouvez venez me demander au cas par cas si vous avez des doutes.

Vous êtes aussi encouragés à regarder les liens sur mon site et stack overflow si vous avez des problèmes. Vous pouvez m'envoyer des emails pour avoir des précisions ou un peu d'aide, mais si une recherche google me donne le résultat dans les 30 secondes ce sera moyennement apprécié.

5 Remarques

La remarque la plus importante est que presque tous les choix sont libres, tant qu'ils sont argumentés. Vous avez donc une très grande liberté, mais éloignez-vous du cahier des charges à vos risques et périls.

5.1 Mode de travail

Je vous conseille fortement de travailler en binôme, en réflechissant à l'avance puis en implémentant chaque fonctionnalité de manière incrémentale et en vérifiant que vous maintenez les fonctionnalités précédentes. Ce projet a l'avantage d'être fortement compatible avec ce mode de développement. Le code à deux, avec un des deux vérifiant que tout marche et donnant des conseils peut être bien plus productif que la séparation en tâches distinctes à répartir. L'utilisation d'un gestionnaire de version est aussi une très bonne idée. Github notamment est gratuit pour les système open-source, tout comme darcs hub.Ce projet devrait prendre quelques dizaines d'heures de vrai travail en intégrant certains bonus. Avec les autres fonctionnalités additionnelles cela peut naturellement augmenter mais la charge de travail reste raisonnable (une codeuse décente pourrait sans doute le faire en 10-20h). Vous pouvez vous inspirer de FileZilla et d'autres logiciels si vous les désirez.

5.2 Questions

Vous avez plusieurs possibilités d'implémentation. Vous devez notamment vous poser les questions suivantes :

- Par quel bout du projet commencer ? (Simple)
- Comment faire un dispositif permettant de tester votre propre logiciel ?
- Quelles sont les différentes manières de copier un fichier ?
- Quelles permissions sont nécessaires ?
- Comment gérer les différentes files d'attentes, et quelles structures de donnée utiliser ?
- Comment savoir combien de temps il reste pour achever un transfert ?
- Comment vérifier qu'un transfert s'est bien déroulé ?
- Comment faire la communication entre processus, et comment standardiser la connection entre client et serveur ?
- Que se passe-t-il quand un client essaye de supprimer un dossier alors qu'un autre essaye d'écrire un fichier dedans ?
- Comment optimiser le serveur quand il a plusieurs clients ?

6 Modalités

Votre note sera déterminée en immense partie par votre projet, mais une rapide soutenance avec démo aura lieu, ou bien in meatspace ou bien via skype. Vous êtes encouragés à vous entraider sans faire de copie de code. Cette soutenance servira donc à vérifier que vous comprenez bien tout ce que vous faites. Vous avez le choix d'être en monôme ou en binôme, avec plus d'indulgence pour les monômes mais un encouragement à faire tout de même du binôme. Les deux membres du binôme devant naturellement chacun comprendre tout le code.
Vous serez notés avec la formule (Contrôle Continu $\times$ 0.35 + Examen $\times$ 0.65), mais le contrôle continu est noté sur plus que 20. Il est donc plutôt facile d'avoir une très bonne note dans ce cours (l'an dernier il y avait 2 élèves à 19 et 20, et 4 de plus entre 14 et 17).

6.1 Rendu

En plus de cette soutenance rapide, j'attendrai de votre part ou bien une archive contenant tous vos fichiers, ou bien un lien vers un github. Il faut qu'il y ait :

- Les fichiers sources
- Un readme
- Un makefile ou assimilé
- Un rapport de projet expliquant vos choix, en pdf (qui peut inclure le contenu du readme).

6.2 Base

- Tout projet ne compilant pas aura 0.
- Un projet implémentant les commandes de base, ne souffrant pas de segfault et remplissant le cahier des charges aura au minimum la moyenne (avant pénalités).

6.3 Pénalités

- Un code commenté et bien écrit est non seulement facile à maintenir mais aussi à relire et comprendre (ce qui est utile quand on a entre 8 et 16 projets à vérifier). Les codes illisibles souffriront d'une pénalité.
- Les erreurs systématiques non gérées donneront lieu à des pénalités.
- De même, l'absence de système de type makefile et de readme sera pénalisée (sauf très bonne justification).
- Une mauvaise gestion des permissions sera légèrement pénalisée.
- Chaque jour de retard par rapport à la deadline vous coûtera un point de plus que le précédent (donc après 6 jours vous avez -1/20).

6.4 Bonus

Les fonctionnalités additionnelles seront fortement récompensées tant que les fonctionnalités de base fonctionnent, de quoi avoir facilement la moyenne même en plantant l'examen final – ce que je ne préfèrerais pas cependant (trois étudiants ont évité le rattrapage l'an dernier grâce à cela). Vous n'avez pas d'accès au barème exact. Des fonctionnalités de votre invention seront jugées selon leur inventivité, utilité, et qualité d'implémentation. D'après mon barême provisoire avoir 50 au projet reste faisable.