M1 EIDD 2016
Projet Systèmes et Réseaux

Enseignant : Nicolas K. Blanchard (prénom.nom@ens.fr)

21 février 2016

Résumé

Le but de ce projet est d'implémenter un système de messagerie interne multi-utilisateurs. 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 Système de messagerie

Le but du projet est d'avoir un système de messagerie sur un système multi-utilisateurs. Tous les utilisateurs du système sont donc connectés à la même machine sans communication réseau. Chaque personne voulant utiliser le système de messagerie possède dans son répertoire d'accueil deux sous-répertoires, correspondant aux messages reçus et envoyés.

Chaque message est composé de quatre parties :

— La première ligne, commençant par "#sujet :", puis le sujet du message et enfin le caractère de nouvelle ligne '\n'.
— La première ligne, commençant par "#a :", suivi de la liste des noms des destinataires, séparés par ", "
— Ensuite vient le corps du message, jusqu'à la ligne commençant par "#liens :\n "
— Chaque ligne suivante contient une référence vers un document attaché.

Le but du projet est d'écrire un système permettant de déposer, lire, et transferer des messages. Un exemple d'ensemble de commandes est présenté dans la partie 2, mais il est possible de faire différemment si votre système est strictement supérieur (en termes de sécurité / d'espace mémoire nécessaire ou de simplicité d'utilisation). La commande recuperer par exemple n'est pas forcément nécessaire.

2 Commandes

2.1 Deposer

La commande deposer permet d'envoyer un message. Cette commande aura la forme suivante :

deposer -m message -a destinataires -s sujet -l liens

Les options -m, -a, -s, et -l pouvant à priori apparaître dans n'importe quel ordre.

— message est une référence vers un fichier qui contient le contenu du message. Il y a au plus une option -m (il est possible d'envoyer des messages sans contenu et dans ce cas il n'y a pas d'option -m).
— destinataires indique les destinataires du message. Chaque destinataire est identifié par son login. C'est le login de la personne qui doit recevoir le message. L'option -a est obligatoire vu qu'il faut au moins un destinataire.
— sujet donne le sujet du message. L'option -s est obligatoire (pas de message sans sujet) et il faut un seul sujet.
— liens est une série de références vers des documents attachés au message envoyé. L'option -l est facultative.

Pour éviter que deux messages s'écrasent, on propose d'utiliser un compteur dans les messages reçus et les messages envoyés.

2.2 Lire

La commande

lire num

affiche le message reçu dont le numéro est num. Il faut aussi afficher le nom de l'expéditeur et la date d'envoi.

2.3 Lister

La commande ”lister” affiche la liste des messages reçus. Elle aura la forme suivante :

lister -t -n nom -i numéro -s string -c string

Chaque ligne de la liste contient :
— Le numéro du message (correspondant au lien symbolique dans le répertoire de réception.
— Le nom de l'expéditeur
— Le sujet du message
— La date d'envoi

Sans options, cette commande affiche la liste de tous les messages reçus. Les options suivantes sont cependant possibles :
— -t, pour ne sélectionner que les messages reçus le jour même.
— n nom n'affiche que les messages envoyés par l'utilisateur nom.
— -i numéro n'affiche que les messages dont le numéro est supérieur à numéro.
— -s string et -c string t n'affichent que les messages dont respectivement le sujet ou le corps de texte contiennent la chaîne de caractère string.

2.4 Recuperer

La commande

recuperer -i num -c chemin

Copie les documents attachés au message num vers le répertoire indiqué par chemin, ou bien le répertoire courant si chemin n'est pas spécifié.

2.5 Transferer

La commande

transferer -i num -a destinataire

transfère le message num à un autre destinataire.

2.6 Initialiser

La commande

initialiser

Initialise le système de messagerie pour un utilisateur, créant les répertoires et fichiers nécessaires.

3 Exemples

Supposons que l'on écrive

deposer -m message.txt -a Samia Emmanuelle Alex -s "Ceci n'est pas un dickpic” -l /usr/home/enorme.jpg ../vids/rickroll.mp4

Cela permet d'envoyer un message à Samia, Emmanuelle et Alex, dont le sujet est "Ceci n'est pas un dickpic”. Le contenu du message est dans message.txt, dans le répertoire courant. Le message a aussi deux liens, qui peuvent être indiqués de manière absolue ou relative. Si l'expéditeur est Stefan qui a lancé le programme depuis /usr/home/mydocs/notporn , cela crée un fichier dans le répertoire envoyes dont le contenu est :

#sujet : Ceci n'est pas un dickpic

#a Samia Emmanuelle Alex

contenu du message copié depuis mes.txt

#liens

/usr/home/enorme.jpg
/usr/home/mydocs/vids/rickroll.mp4

4 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.

4.1 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. Petite information, pour un utilisateur moyennement compétent et inexpérimenté ne sachant pas vraiment faire de C (votre enseignant en 2011), ce projet prend une dizaine d'heures de vrai travail avec quelques bonus. Avec les autres fonctionnalités additionnelles cela peut naturellement augmenter mais la charge de travail reste raisonnable.

4.2 Architecture

Vous avez plusieurs possibilités d'implémentation. Vous devez notamment vous poser les questions suivantes :
— Comment éviter la multiplication (recopiage) des données quand un utilisateur veut envoyer un fichier massif à de nombreux autres utilisateurs ?
— Que se passe-t-il si plusieurs utilisateurs essayent d'envoyer un message au même moment ?
— Comment gérer les droits à l'intérieur des répertoires recus et envoyés ?
— Comment bien gérer les options multiples et facultatives dans le désordre ?
— Est-ce que l'utilisation d'un fichier message.txt est vraiment nécessaire ?

4.3 Commandes

Deposer

Pour cette commande vous devrez faire attention à la fois aux permissions et à l'écriture simultanée par plusieurs utilisateurs de messages. Des mutex sont donc à priori nécessaires (ou des sémaphores pour celles qui veulent). De même, il faut faire attention aux exceptions : que faire si le destinataire n'existe pas, ou si le fichier message n'est pas à la l'emplacement indiqué ?

Lister

Cette commande possède de très nombreuses options que vous pourriez implémenter, comme la recherche parmi un ensemble de dates, la détection de liens périmés et leur suppression ou d'autres de votre choix.

Initialiser

Vous pouvez avoir envie de faire des choix permettant une meilleure gestion des droits en demandant que la fonctionnalité initialiser ne soit utilisable qu'avec des droits supérieurs. Vous pouvez aussi avoir envie de cacher certains répertoires ou fichiers (comme le compteur de numéro). Pensez-y.

5 Fonctionnalités additionelles (Bonus)

En plus des choix alternatifs d'architecture et de commandes, vous pouvez aussi contempler les possibilités suivantes :
— L'implémentation d'une interface graphique sommaire (avec les librairies adaptées, votre prof n'étant pas sadique non plus).
— Une réflexion poussée sur la possibilité d'un système de ce type coordonné en réseau sur plusieurs machines.
— La réalisation d'un tel système multi-machines multi-utilisateurs.
— Des commandes de recherches plus poussées (qui évitent de relire l'intégralité des messages pour trouver une chaîne de caractère par exemple).
— Une bonne gestion des exceptions, et une aide intégrée au logiciel sont aussi des bonnes idées.
— Enfin, d'autres commandes, pour supprimer des messages (seuls ou en masse) par exemple, sont implémentables.

6 Modalités

Votre note sera déterminée en immense partie par votre projet, mais une rapide soutenance avec démo aura lieu. 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.
Enfin, plusieurs TPs notés auront lieu sur des problématiques de réseau formeront le reste de votre contrôle continu. Vous serez notés avec la formule $(\text{Contrôle Continu} \times 0.35 + \text{Examen} \times 0.65)$, mais le contrôle continu sera noté sur plus que 20. Il est donc plutôt facile d'avoir une très bonne note dans ce cours.

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 ne supprimant pas les messages dans un cadre multi-utilisateurs 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.
- 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.

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. Vous n'avez pas d'accès au barème exact mais la question théorique pour le réseau et les autres questions d'architecture vaudront entre 1 et 2 points. Chaque fonctionnalité additionelle implémentée vaudra à priori jusqu'à 5 points (pour les grosses comme le réseau ou le gui), selon la qualité naturellement. Des fonctionnalités de votre invention seront jugées selon leur inventivité, utilité, et qualité d'implémentation.

7 Outils

Vous avez le droit aux librairies de votre choix tant que vos dépendances sont bien gérées et justifiées, et que votre projet compile de manière indépendante. Une liste de fonctions dont vous pouvez avoir (fortement) besoin :

lstat, getpwuid, time, localtime, fcntl, getcwd, getpwnam, getenv...

Vous êtes aussi encouragés à regarder les liens sur mon site et stack overflow si vous avez des problèmes. Vous pouvez aussi 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.