Projet de Programmation Système

Juliusz Chroboczek

20 octobre 2013

Introduction

Une fonctionnalité commune des réseaux sociaux est le « mur » ou « Dazibao », une structure de données servant à partager des commentaires, des images etc. Typiquement, un Dazibao peut être écrit par un utilisateur ou par un groupe d'utilisateurs, et peut être lu par un groupe d'utilisateurs ou par tout le monde.

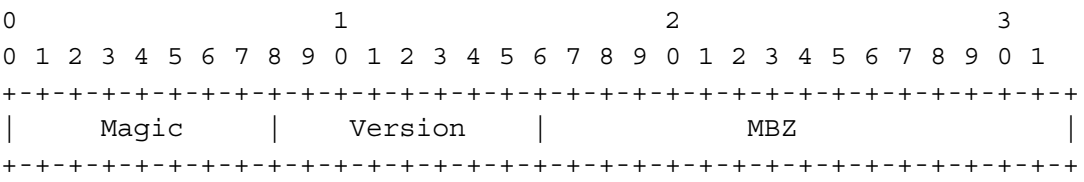
Le but de ce projet est d'implémenter une collection de programmes qui permet de manipuler un Dazibao représenté par un fichier dans un système de fichiers partagé (paragraphe 2) et un programme qui permet d'être notifié des changements apportés à un tel Dazibao (paragraphe 3).

1 Structure du Dazibao

Un Dazibao est représenté par un fichier binaire consistant d'un entête de quatre octets suivi d'un nombre arbitraire de triplets « Type-Longeur-Valeur » (TLV). L'entête contient une signature permettant d'identifier les Dazibao et un numéro de version ; les TLV contiennent chacun une donnée.

1.1 Détail de l'entête

L'entête consiste de quatre octets : un numéro magique d'un octet, un numéro de version d'un octet, et deux octets réservés à des extensions futures. Sa structure est représentée par le diagramme suivant :



Champs :

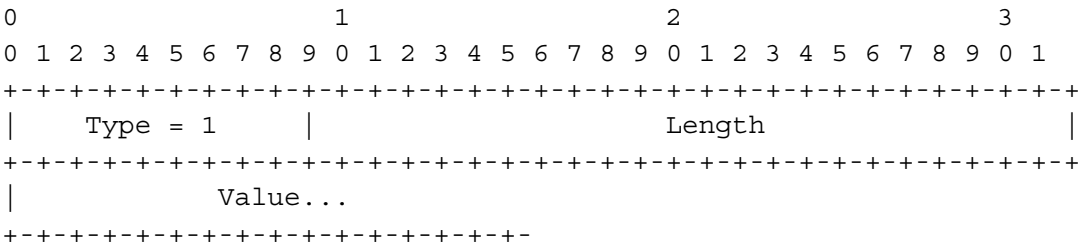
- *Magic* : ce champ vaut 53 ; votre programme devra rejeter tout fichier qui ne commence pas par cette valeur.

– *Version* : ce champ vaudra 0 ; votre programme devra rejeter tout fichier dont le second octet ne vaut pas 0.
– *MBZ* : ce champ est réservé pour des extensions futures ; il vaudra normalement 0, mais votre programme devra l'ignorer.

1.2 Détail des différents TLV

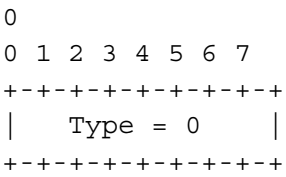
À l'exception de *Pad1*, tous les TLV consistent d'un octet qui indique le type du TLV, de trois octets qui indiquent la longueur de la valeur (à l'exclusion des champs *Type* et *Length*, et d'un nombre variable d'octets contenant la valeur.

Cette structure est représentée par le diagramme suivant :



1.2.1 Pad1

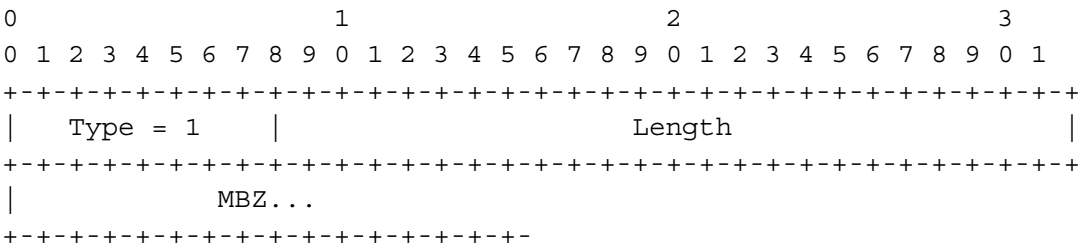
Un TLV *Pad1* consiste d'un seul octet ; il est toujours ignoré (i.e. il est invisible pour l'utilisateur de votre programme).



Champs :
– *Type* : vaut 0, ce qui indique un TLV *Pad1*.

1.2.2 PadN

Un TLV *PadN* consiste d'un nombre variable d'octets (au moins deux) qui sont toujours ignorés.

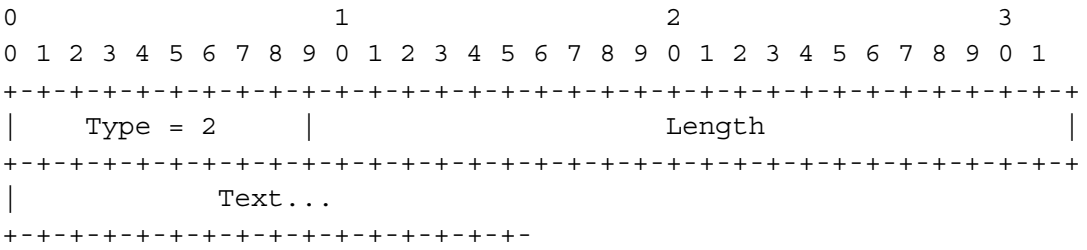


Champs :

- *Type* : vaut 1, ce qui indique un TLV *PadN*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *MBZ* : ce champ sera toujours écrit comme une suite de zéros binaires, et ignoré lors de la lecture.

1.2.3 Text

Un TLV *Text* contient un fragment de texte Unicode.

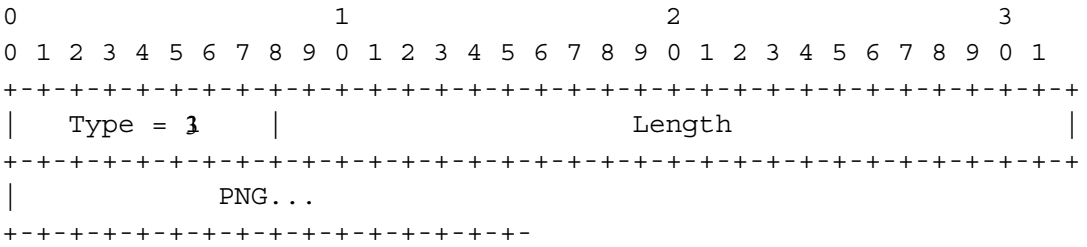


Champs :

- *Type* : vaut 2, ce qui indique un TLV *Text*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *Text* : un texte arbitraire en codage UTF-8, sans octet nul terminal, de longueur *Lengtht*.

1.2.4 PNG

Un TLV *PNG* contient une image au format *PNG*.

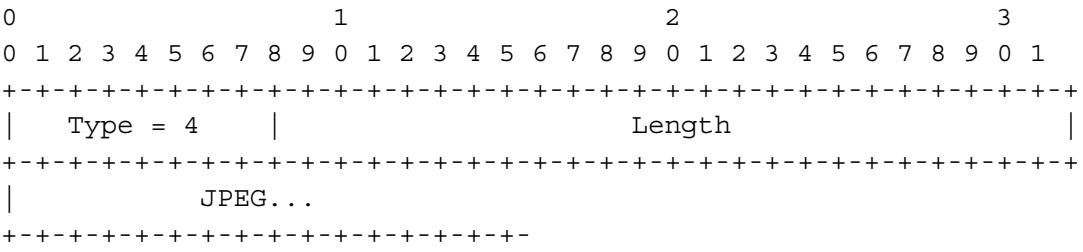


Champs :

- *Type* : vaut 3, ce qui indique un TLV *PNG*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *PNG* : une image au format *PNG*, de taille *Length*.

1.2.5 JPEG

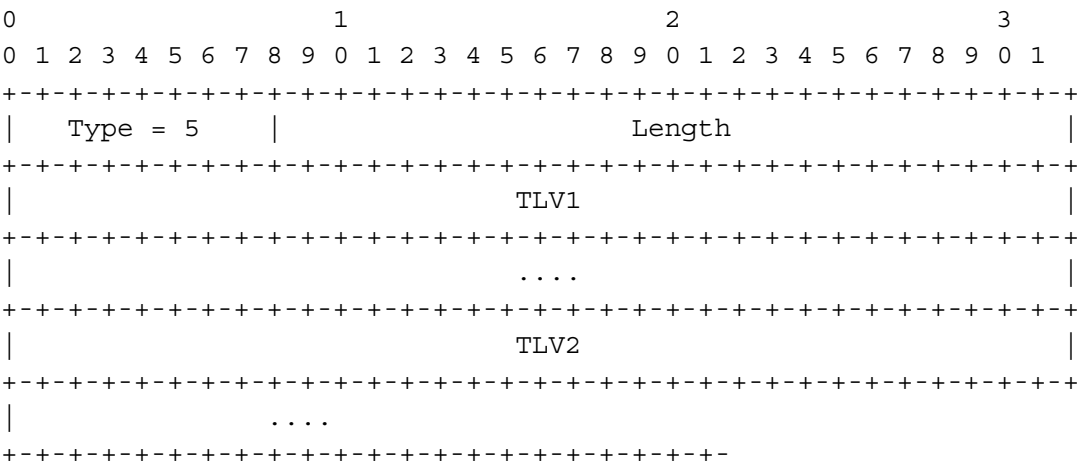
Un TLV *JPEG* contient une image au format *JPEG*.



Champs :
- *Type* : vaut 4, ce qui indique un TLV *JPEG*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *JPEG* : une image au format *JPEG*, de longueur *Length*.

1.2.6 Compound

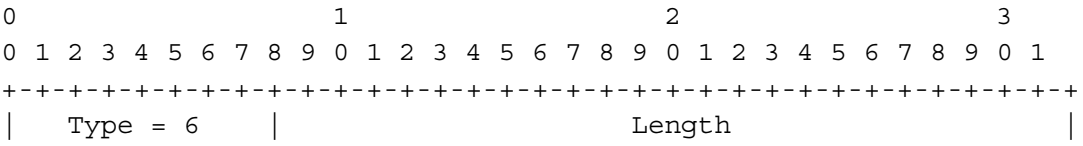
Un TLV *Compound* contient une suite de TLV de types arbitraires qui sont manipulés comme une seule unité.

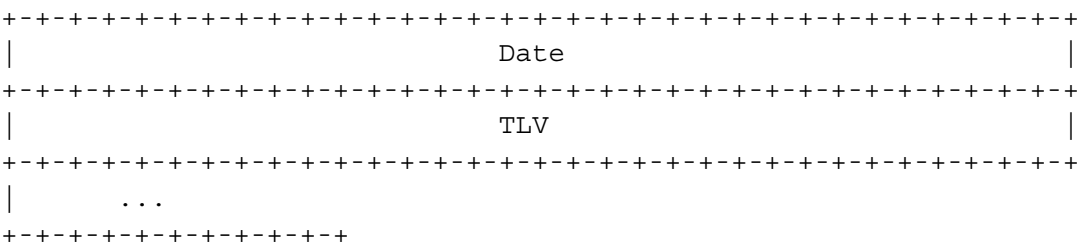


Champs :
- *Type* : vaut 5, ce qui indique un TLV *Compound*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *TLV1*, *TLV2*,… : une suite de TLV, dont la longueur totale est égale à *Length*.

1.2.7 Dated

Un TLV *Dated* contient une date et un TLV de type arbitraire.





Champs :

- *Type* : vaut 6, ce qui indique un TLV *Date*.
- *Length* : la longueur en octets de la valeur du TLV (à l'exclusion des champs *Type* et *Length* ;
- *Date* : la date attribuée au TLV, représentée comme un nombre de secondes UTC depuis le premier janvier 1970, et codée comme un nombre de 32 bits en format gros-boutiste.
- *TLV* : un TLV de type arbitraire.

1.3 Mécanisme d'extension

Comme tous les TLV ont le même format, un programme est capable d'ignorer un TLV de type inconnu, ce qui permet d'étendre le format de fichier. Cependant, quelques règles minimales sont à respecter.

Les types de TLV 7 à 127 sont réservés pour les extensions « officielles » au format. Si vous vous en servez sans mon autorisation, je tue un châton.

Les types de TLV 128 à 255 sont réservés pour les extensions faites par les étudiants. Vous pouvez vous en servir pour n'importe quel usage, du moment que vos TLV ont le format général décrit au début de cette section.

2 Programme de lecture et manipulation du Dazibao

Nous vous demandons de nous fournir un programme capable d'afficher le contenu d'un Dazibao, d'ajouter une entrée à la fin d'un Dazibao, de supprimer une entrée d'un Dazibao, et de compacter un Dazibao.

L'interface utilisateur est laissée à votre choix. Vous pourrez par exemple implémenter un ensemble d'utilitaires en ligne de commande, un programme présentant un menu textuel, une interface graphique, ou (pourquoi pas ?) une application web.

2.1 Lecture

Pour lire un Dazibao, un programme commence par ouvrir le Dazibao en mode lecture seule, et prend un verrou *flock* partagé sur le fichier[1]. Il vérifie ensuite les champs *Magic* et *Version* de l'entête, et quitte avec une erreur s'ils n'ont pas les valeurs attendues. Il parcourt ensuite la suite de TLV, affichant chacun d'entre eux au passage.

S'il rencontre un TLV d'un type inconnu, votre programme devra afficher un message à cet effet, puis correctement lire les TLV qui suivent. Si votre programme fonctionne en mode texte, il

1 Si vous utilisez un verrou Système V *(fcntl)*, je vous trouverai où que vous soyez.

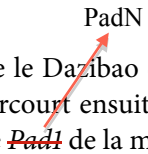
pourra par exemple afficher un message de la forme « Image PNG » ou « Image JPEG » lorsqu'il rencontre un TLV de type 3 ou 4.

2.2 Ajout

Pour ajouter une entrée à un Dazibao, un programme ouvre le Dazibao en mode lecture et écriture, puis prend un verrou *flock* exclusif sur le fichier. Il ajoute ensuite le nouveau TLV à la fin du fichier.

Une optimisation est possible : si le Dazibao se termine par un TLV de type *Pad1* suffisamment grand pour contenir le nouveau TLV, il est possible de stocker le nouveau TLV à la place de ce dernier.

2.3 Suppression



Pour supprimer une entrée d'un Dazibao, un programme ouvre le Dazibao en mode lecture et écriture, puis prend un verrou *flock* exclusif sur le fichier. Il parcourt ensuite le fichier pour trouver le bon TLV à supprimer, et le remplace par un TLV de type ~~*Pad1*~~ de la même taille.

Des optimisations sont possibles ; en particulier, si le TLV supprimé est précédé ou suivi d'un TLV de type *Pad1* ou *PadN*, il est possible de fusionner les enregistrements. De plus, si l'enregistrement supprimé est le dernier du Dazibao, il est possible de tronquer le fichier. Prenez garde, cependant, les examinateurs préfèrent un programme non-optimisé correct à un programme optimisé mais faux.

2.4 Compaction

Après plusieurs opérations, votre Dazibao contiendra plusieurs enregistrements de type *Pad1* ou *PadN*, qui occuppent inutilement de la place. La *compaction* consiste à éliminer ces enregistrements, à décaler les TLV qui suivent, et finalement à tronquer le fichier résultant.

Votre programme devra être capable de compacter un Dazibao en un temps au plus proportionnel à la taille du fichier d'origine.

3 Notifications

La plupart d'entre nous consultent les Dazibao de leurs amis au plus une fois par jour, typiquement le matin, en arrivant au bureau. Certains, cependant, aiment être notifiés en temps réel des modifications apportées aux Dazibaos qui les intéressent.

Nous vous demandons donc de nous fournir un programme qui prend en paramètre une liste de Dazibao, et crée une *socket* de domaine Unix qui s'appelle `.dazibao-notification-socket` dans votre répertoire *home*. Lorsque l'un des Dazibao est mis à jour, votre programme devra écrire à tous les clients de la *socket* un message de la forme

```
C/home/jch/Dazibao
```

Le caractère `C` indique qu'un changement a eu lieu, et la suite de la ligne indique le chemin du Dazibao qui a changé.

4 Détails d'implémentation

Ce projet laisse beaucoup de détails à votre choix ; nous nous autoriserons à prendre en compte vos choix d'implémentation en compte lors de l'évaluation

Accès concurrents Votre programme devra être capable d'accéder au fichier concurremment à d'autres programmes — il faudra absolument utiliser les verrous. L'utilisation incorrecte des verrous sera fortement pénalisée.

TLV inconnus Votre programme devra être capable de lire des fichiers contenant des types de TLV inconnus, et d'afficher les TLV qui suivent un TLV inconnu.

Interface utilisateur Comme mentionné dans le paragraphe 2, le choix de l'interface utilisateur est laissé à votre choix. Vous pouvez par exemple nous fournir un ensemble de commandes à utiliser à partir du *shell*, ou un programme avec une interface texte, ou un programme avec une interface graphique, ou une inteface *web*, ou un type d'interface auquel je n'ai pas pensé. Bien-sûr, la qualité de l'interface utilisateur sera prise en compte lors de l'évaluation.

Accès au fichier Il existe plusieurs façon d'accéder au fichier. Vous pouvez par exemple utiliser la bibliothèque *stdio*[2], vous pouvez utiliser les appels système `read` et `write`, vous pouvez utiliser `readv` et `writev`, ou vous pouvez faire des lectures ou des écritures dans une zone mémoire mappée par `mmap`[3]. De même, pour détecter les changements apportés à un Dazibao, vous pouvez périodiquement lire le Dazibao dans son intégralité, vérifier sa date de dernière modifiction, ou utiliser une API non-portable de notification.

Toutes ces techniques vont fonctionner, mais certaines sont plus efficaces ou plus élégantes que d'autres. *Caveat programmator.*

5 Extensions

Toutes les extensions seront les bienvenues, et seront prises en compte lors de l'évaluation. Si vos extensions définissent de nouveaux types de TLV, elles devront *obligatoirement* obéir au mécanisme d'extension décrit dans le paragraphe 1.3.

Si vous êtes à court d'idées, voici quelques idées d'extension.

Opérations supplémentaires Dans le paragraphe 2, nous vous décrivons un certain nombre d'opérations que votre programme doit obligatoirement implémenter. Si vous avez une interface utilisateur plus riche que la mienne, vous aurez sans aucun doute envie d'implémenter d'autres opérations, par exemple l'édition d'une entrée.

Types de données supplémentaires Une façon de stocker les films dans le Dazibao ? Les modèles 3D ? Les formules mathématiques ?

2 Attention, alors, à ne pas relâcher le verrou avant d'avoir vidé les tampons.

3 Attention alors aux écritures asynchrones et leur interaction avec les verrous — `msync` est votre ami.

Automatisation Certains exhibitionnistes aiment bien faire des notifications automatisées sur leur Dazibao — « Pierre a lu son mail », « Jacques a compilé un programme C ce qui a généré 12 erreurs » ou « Paul s'est brossé les dents ». Votre implémentation est-elle automatisable (par exemple à partir d'un *script*) ? Fournissez-nous un exemple de programme qui fait des notifications automatiques.

Logiciel de notifications Le logiciel de notification que nous vous fournissons est très primitif. Écrivez un remplacement pour ce programme qui est plus joli, plus pratique, ou alors qui s'intègre au système de notifications de *Gnome*.

Autres extensions Mettez nous en plein la vue.

6 Soumissions

Le projet sera à traiter en groupes de 3 personnes (les projets soumis par des groupes de 4 personnes ou plus ne seront pas acceptés). Votre solution devra consister d'un programme écrit en C (compilable et utilisable sous Linux) accompagné de :

- un fichier texte nommé README indiquant brièvement comment compiler et utiliser votre programme ;
- un rapport sous format PDF de 2 à 8 pages environ décrivant sommairement votre soumission, décrivant les extensions traitées, et expliquant (ou justifiant) les choix de conception ou d'implémentation que nous pourrions ne pas comprendre du premier coup ;
- tout autre fichier nécessaire à la compilation et à l'exécution, par example un fichier `Makefile`, un script permettant de compiler vos sources, etc.

Votre soumission devra obligatoirement être soumise par courrier électronique à `jch@pps.univ-paris-diderot.fr` ; la date limite sera indiquée sur la page *web* de l'auteur[4]. Le courrier que vous m'enverrez devra *obligatoirement* avoir un entête `Subject` contenant la chaîne « `Projet système` », et contenir une archive `tar.gz` en attachement.

4 http ://www.pps.univ-paris-diderot.fr/ jch/enseignement/systeme/