Comment je traite vos mails...

Nicolas Bareil (nbareil@madchat.org)

9 octobre 2002

Ce papier a pour but de vous faire partager mes idées vis-à-vis du traitement du mail, en effet, vous trouverez ici toutes les étapes que doit franchir un mail avant sa réception réelle. C'est un véritable parcours du combattant : filtres, *anti-spamming*, tris, ajout d'en-têtes, classements...

On ne se limitera pas à ces seules étapes, mais également à la configuration de mon logiciel de lecture de mail. Enfin, pour boucler le cycle de vie d'un mail, nous aborderons le sujet de l'archivage, puis de la sauvegarde du courrier. Je vous rassure néanmoins, aucun mail n'a été maltraité ou brutalisé lors de l'écriture de ce papier.

Pourquoi être aussi attaché aux mails et détester autant les lieux de discussion tels que *IRC*, ou les messageries instantannées ? Voici les principales raisons :

Difficultés de compréhension Si vous ne vous appellez pas *Richard M. Stallman*[1] ,lorsque vous écrivez un mail, vous pouvez prendre votre temps avant de répondre alors que lors d'une discussion orale, ou sur une messagerie instantanée, il est facile d'écorcher un nom de variable, ou tout simplement de dire des bêtises...

Pérennité des données On n'a plus le problème de la mémoire qui flanche : tout est classé, archivé, puis sauvegardé sur un média adéquat. Cela permet de retrouver des éléments très facilement, en particulier si on fait régulierement intervenir un moteur d'indexation, tel que swish++.

Facilité d'accès Vous cherchez une information trouvée sur IRC, ou lors de bavardages ? Bonne chance, à moins d'avoir une très bonne mémoire. Qu'importe la facon dont vous classez vos mails, vous retrouverez toujours votre réponse, même si ça vous entraine à faire des `grep -r mysearch Mail/` qui durent plusieurs minutes.

Intégrité des données Avec l'usage de GnuPG, un mail peut désormais être signé, voir chiffré en cas de besoin. Cela vous assure donc une parfaite authentification de l'émetteur, ainsi qu'une certaine confidentialité.

Lecture aisée Ecrire du code, voire répondre à une question technique est un exercice assez délicat sur IRC : soit quelqu'un vous coupe au milieu de votre élan, soit votre client IRC supprime toute votre indentation, soit certains éléments sont supprimés (car le client IRC interprète vos données).

[1] Je prends l'exemple de RMS car il est connu pour le nombre impressionnant de mails auquels il répond

Chapitre 1

Traitement du mail en amont

1.1 Adresses spécifiques

1.1.1 Principe

Il est préférable de ne pas tout mélanger : mails personnels, mailing-lists, et adresses poubelles. Cette dernière est destinée à être donnée lors de l'inscription à des portails. Car on évite alors les *newsletters*, ou autres *spams*. Pourquoi distinguer les boîtes aux lettres ? Par exemple, lorsque vous n'êtes pas sur votre machine personnelle, et que vous consulter vos mails à partir d'une interface *web*, vous vous rendrez très rapidement compte qu'il n'est pas très agréable de naviguer entre toutes les pages pour trouver le mail de votre copine noyé entre une centaine de messages de listes de diffusion.

Une boîte destinée aux mailing-lists peut facilement recevoir jusqu'à 400 mails par jour, donc mélanger ces derniers avec les courriers personnels serait une erreur. Car impossible à trier manuellement.

1.1.2 En pratique

Comme mentionné ci-dessus, je possède plusieurs adresses :

nbareil@matoufou.org C'est l'adresse par défaut pour les mails privés, c'est celle-ci que j'utilise le plus souvent. En fonction du destinataire, cela peut aussi être *offset@madchat.org*, *nbareil@madchat.org*, ou tout simplement *nbareil@free.fr*.

Les mails arrivant sur cette adresse ont une priorité élevée : aussi bien lors de la récupération des mails que lors du filtrage, puis lors de la lecture.

nbareil.ml@matoufou.org Adresse pour les mailing-lists, c'est elle qui va recevoir l'intégralité des différentes listes que je reçois, son volume est de plusieurs centaines de mails par jour (il y a en effet la majeure partie des ML de debian.org). Un mail personnel envoyé à cette adresse sera forcément lu, mais avec un peu de retard. Il n'est pas sûr du tout qu'il franchisse les filtres, car considéré comme *spam*.

spam@matoufou.org Lors d'une inscription à un site internet, je fournis cette adresse, mais je ne la relève pratiquement jamais, car elle ne contient que

du courrier non-solicité. Un mail privé qui arriverait là ne serait pas lu. C'est certain.

admin@domaine.org Tous les domaines (voire serveurs) que j'administre possèdent tous une adresse du type admin@ qui est indépendante des autres serveurs. En cas de panne, compromission, ou problème, c'est vers cette adresse que les *logs* iront.

1.2 Premier filtrage, lors de la transaction SMTP : ORDB

En règle générale, mes adresses ne sont que des *alias* vers des adresses du type xxxx.dyndns.org, xxxx.d2g.com...Ce sont des serveurs où j'ai un accès super-utilisateur (*root*). Pourquoi cette condition obligatoire ? Car je veux avoir la certitude que mon traffic mail ne sera pas détourné/espionné. Oui, tout à fait, c'est ce qu'on appelle de la paranoïa. De plus, il est souvent nécessaire que je mette les mains dans les fichiers de configuration du *Mail Transport Agent (MTA)* pour régler quelques problèmes, ce qui demande les privilèges de *root*.

Mais revenons-en au filtrage, celui-ci est fait à partir du MTA en consultant les serveurs RBL, qui bannissent certains domaines suspectés de *spamming*. Cette première étape permet de supprimer une bonne partie des *spams* asiatiques (je n'ai pas de correspondant asiatique, donc je ne risque pas grand chose en cas de *blacklistage* abusif). J'utilise le service fournit par ordb.org

Une fois cette étape réalisée, le mail va passer par une série de filtres *Postfix*[1], ceux-ci peuvent être trouvés ici[2]. Ce sont des filtres simples, mais comportant des expressions (rationnelles) typiques d'un *spam*. Postfix est également configuré pour refuser les mails non-conformes aux normes (absence de certains *headers*, ou remplis d'une façon incorrecte...) seront refusés sans aucune autre forme de procès. À ce propos, je ne vais pas tarder à refuser les mails en HTML seulement (mais autoriser les mails présentant une alternative text/plain). Vous voilà prévenu les *gorets*...

1.3 Le filtrage continue

Les mails sont ensuite passer à procmail [3],qui va s'occuper de faire du filtrage encore une fois. le courrier va alors passé par *Razor*[4]. En quelques sortes,*Razor* est une grosse base de donnée (libre d'accès) contenant l'ensemble des spams qu'on lui a signalé. Le client, sur notre machine, va alors contacter ce serveur et lui donner un identifiant unique du mail qu'on vient de recevoir. Si ce mail est connu de la base Razor, c'est que c'est sûrement un *spam*. Je dis sûrement, car il y a parfois des *falses-positives*, un exemple est les *Debian Weekly News* qui sont à chaque édition rapportées comme *spam*. Néanmoins, cela reste rare.

Si le mail n'a pas été éliminé, il passe alors dans le populaire *spamassassin*[5], celui-ci va alors effectuer des centaines (des milliers ?) de tests sur les *headers* du

[1] Postfix : http ://www.postfix.org/
[2] Filtres Postfix : http ://http ://colino.net/computing/unix.php3
[3] Procmail : http ://www.procmail.org/
[4] Razor : http ://razor.sourceforge.net/
[5] Spamassassin : http ://www.spamassassin.org/

mail. Par exemple, s'il voit des dollars ($$$), il va assigner un certain score au mail, et si à la fin de tous les tests, ce score atteint une valeur seuil, le courrier sera considéré comme *spam* (en ajoutant un *header* que l'on classera grâce à Procmail).

Voici les règles à mettre dans votre `~/.procmailrc` :

```
# ~/.procmailrc

SPAMBOX=$MAILDIR/caughtspam

:0 Wc
| razor-check
:0 Waf
| formail -A "X-Razor-Warning: SPAM."

:0:
* ^X-Razor-Warning: SPAM
$SPAMBOX

# spam in trash
:0fw
| spamc

:0e
{
 EXITCODE=$?
}

:0:
* ^X-Spam-Status: Yes
$SPAMBOX
```

En tout, le mail aura passé plus de quatre filtres (j'en ai volontairement éludé quelques un pour simplifier). Malgré tout ce système, il y a quand même quelques spams qui passent au travers des mailles du filet (en majorité des spam francophones) . Mais cela reste en dessous de cinq messages par mois, pour information, sans filtre, j'atteignerai la centaine par mois. Je peux donc considérer le système comme efficace.

Dans un futur plus ou moins proche (traduire par : dès que j'aurai le temps), je configurerai un serveur supportant le système *TMDA*[6], ce système, qui semble relativement casse-pied à installer, ne repose pas sur un principe passif comme tous les autres moyens de filtrage présentés ci-dessus. En effet, à la réception d'un mail d'un expéditeur inconnu, TMDA va lui envoyer un message demandant de confirmer l'envoi de ce message, généralement, un *spam* ne contient pas d'adresse expéditeur valide, donc, en l'absence de confirmation, le mail sera considéré comme *spam*. Le système comporte de nombreuses autres fonctionnalités, mais comme je ne l'ai pas encore testé, je ne peux pas vous en dire plus malheureusement.

[6]TMDA : http ://www.tmda.net/

Chapitre 2

Rapatriement, classement, et lecture du mail

2.1 Rapatriement

Étant donné que je suis encore en RTC (56K), je ne suis pas connecté en permanence, c'est pour cette raison que tout le traitement en amont est effectué sur des serveurs ne m'appartenant pas (Pour la majeure partie, ce sont des lignes ADSL). Il faut donc bien que je rapatrie mes mails. Pour cela, sur les serveurs, j'installe un serveur POP3[1] Je n'utilise pas IMAP car je n'en ai pas une très grande utilité : je n'utilise qu'un ordinateur (N'ayant pas d'accès extérieur au travail) donc pas la peine de faire du zèle avec de l'IMAP, de plus, ce type de serveur consomme énormément de ressources. Mon serveur POP3 favori est celui de *Qualcomm* : *Qpopper*[2]. Ce choix est purement arbitraire, je n'ai fait aucune étude de cas : je l'ai essayé, et ça a marché, donc je le garde pour le moment, à défaut de reproches le concernant.

Niveau client, j'ai longtemps utilisé le programme d'ESR [3] *fetchmail*, mais je n'en suis pas totalement satisfait, je trouve (à tort ?) que c'est une usine à gaz, ensuite, le programme est capable de perdre des mails dans des conditions particulières ; de plus, il ne respecte pas la politique de descripteur de fichier en utilisant *stdout* au lieu de *stderr* comme il le devrait, ce qui est embêtant. Raison supplémentaire, je n'aime pas beaucoup son auteur. Mais cela est complétement subjectif. Ne parlons même pas du fichier de configuration tiré par les cheveux (attention, je n'ai pas dit qu'il était compliqué, juste qu'il était...spécial). En effet, contrairement aux désirs d'ESR qui souhaitait une syntaxe proche d'une phrase en anglais, il en résulte une catastrophe.

J'ai donc, au bout de deux années, préféré utiliser *getmail*, écrit en *Python* par un mec très sympa et très ouvert. Son oeuvre est très légere, et rudement efficace. Le seul regret que j'ai vis-à-vis de ce programme est que son auteur refuse (pour des raisons très justes d'ailleurs) d'y intégrer un mode *daemon*, ce qui nous oblige à lancer le programme toutes les cinq minutes. La solution est donc d'utiliser une tache cron (qui est conçue pour ce type d'utilisation). La

[1] POP3 : Post Office Protocol v3, voir sa RFC pour plus d'infos
[2] Avec le paquet Debian du même nom
[3] Eric S. Raymond, auteur de La Cathédrale et le Bazar

syntaxe du fichier de configuration est beaucoup plus traditionnelle que celle de *fetchmail*. Vous pouvez voir mon fichier ici[4].

Pour ne pas que ma connexion *RTC* s'étouffe, j'empêche *getmail* de rapatrier les mails d'une taille supérieure au mégaoctet. Lorsque cela arrive, le programme va envoyer un *bounce*[5] à l'expéditeur. Pour profiter de cette fonctionnalité, vous devez avoir cela dans votre configuration :

```
# Do NOT download big mails
max_message_size = 1048576
```

Malgré la possibilité de classer ses mails directement par *getmail*, je préfère déléguer ce travail à mon *MDA*[6], favori, en l'occurence, c'est *procmail* qui se charge de cela. Pourquoi lui et pas *getmail* ? Car comme le veut la philosophie Unix, une tâche correspond à un programme. De plus, cela me permet de passer facilement de fetchmail/getmail si j'en ai envie. Autre avantage, si on a un vieux fichier au format *mbox*) que l'on veut reclasser,il suffit de coupler *procmail* à *formail*. La preuve en image :

```
# Fait passer tous les mails contenus dans le fichier
# spool par procmail. Celui-ci va alors classer en fonction
# de nos recettes.

nbareil@mouarf:~% formail -s procmail < spool
```

Cette astuce m'a sauvé la mise plus d'une fois lors de mon changement d'ordinateur.

2.2 Classement par Procmail

2.2.1 Pourquoi procmail ?

Il est vrai que choisir procmail peut paraître étrange, car son code source est très sale, vieillissant, et pas vraiment performant. De plus, son fichier de configuration est également très spécial. Même si les alternatives[7] existent, et rivalisent avec ce dernier, il est très rare que l'on puisse les utiliser sur des machines qui ne sont pas les notres : les administrateurs sont très réticents à installer des logiciels inconnus. Malgré tout, procmail est considéré comme la référence pour faire ce type de tâche. Et les moeurs ne sont pas prêtes de changer.

2.2.2 C'est parti

Mon fichier de règles (Cela donne *rules* en anglais dans le texte) est, comme d'habitude, disponible sur mon site personnel[8]

Ma première règle supprime les doublons, par exemple, lorsqu'une personne me répond depuis une mailing list, il très fréquent que celle-ci m'envoie une copie

[4] Mon fichier de configuration getmail : http ://matoufou.org/linux/
[5] *bounce* : Message d'erreur
[6] MDA : Mail Delivery Agent
[7] maildrop, mailfilter...
[8] Mon .procmailrc : http ://matoufou.org/linux/

(en *Carbon Copy*), or, vu que je lis la liste de diffusion, je le recois deux fois. Un autre avantage intervient lorsque vous interrompez *(fetch/get)mail* alors qu'il récupére des mails. Lorsque vous allez le redémarrer, il va retélécharger tous les mails, même ceux qu'il a déjà prit. Vous vous retrouvez alors avec plein de doublons. La ligne suivante empêche ce phénomène :

```
# ~/.procmailrc

PROCDIR=$MAILDIR/procmail

:0Wh: .msgid.lock
| formail -D 8192 $PROCDIR/msgid.cache
```

Je traite maintenant les listes de diffusion, attention, l'ordre des règles est très important (à moins que vous modifiez les *flags*) : Il faut traiter en premier les listes, puis les messages personnels.

2.2.3 Listes de diffusion

Tout d'abord, il y a le cas particulier des listes du projet Debian : À la fin de chaque message, le serveur rajoute une signature indiquant les modalités pour se désabonner. Ce comportement est très lourd à force : à lire, cela est irritant, et mine de rien, cela prend de la place, faîtes, le calcul, par exemple, sur mes 50 000 mails archivés, la bannière Debian me prend $50000 * 143 = 7150000$. Soit 7 Mo de textes inutiles ! Jacques L'helgoualc'h a écrit un *script* en *AWK* qui supprime la signature Debian. Son script peut-être appliqué sur un fichier *mbox* (mais cela reste déconseillé) ou à la réception de mails. Le programme peut-être trouvé ici : http ://lhh.free.fr/pub/wash-debian.awk, et il vous faut alors mettre ceci dans votre fichier .procmailrc :

```
# ~/.procmailrc

:0 wbf
 * ^X-Loop: *debian
 * < 100000
 |$HOME/bin/wash-debian.awk
```

Etant donné que je suis inscrit à un nombre relativement important de *mailing-lists* (Je suis inscrit actuellement à 78 ML avec un traffic plus ou moins important suivant la liste), j'en avais marre d'avoir à chaque nouvelle liste à ajouter une règle de filtrage dans *procmail*, ou dans *Gnus*. Après quelques recherches, j'ai trouvé cet ensemble de règles écrit par *Nick Moffitt* :

```
# ~/.procmailrc

LISTDIR=$MAILDIR/lists

:0:
* ^Sender: owner-\/[^@<>]+
```

```
$LISTDIR/$MATCH

:0:
* ^X-BeenThere: \/[^@<>]+
$LISTDIR/$MATCH

:0:
* ^Delivered-To: mailing list \/[^@<>]+
$LISTDIR/$MATCH

:0:
* ^X-Mailing-List: <\/[^@<>]+
$LISTDIR/$MATCH

:0:
* ^X-Loop: \/[^@<>]+
$LISTDIR/$MATCH

:0:
* ^List-Id: <\/[^@<>]+
$LISTDIR/$MATCH
```

Grâce à cette recette, *procmail* va trier automatiquement les listes de diffusion. Par exemple, la liste *debian-legal@lists.debian.org* ira directement dans le fichier *$MAILDIR/debian-legal*. Depuis que j'utilise ce système, je n'ai constaté aucun problème.

2.2.4 Mails personnels

En fait, il existe tout de même un problème avec la solution précédente, ces règles vont vous rendre extrêmement fainéant. Et ajouter un filtre sur chaque ami peut vous ennuyer. J'ai donc cherché une solution similaire, mais cela ne marche que pour mes contacts : chaque personne ayant une adresse en madchat.org, matoufou.org, ou madloutre.org va alors avoir droit à une boîte aux lettres automatiquement créé suivant le principe que l'on a vu. Voici mes règles :

```
# ~/.procmailrc

PRIVDIR=$MAILDIR/private

:0:
* ^From:.*@(matoufou|madchat|madloutre)\.org
* ^From:.*[<(]\/[^@()+<>- ]+
$PRIVDIR/$MATCH
```

L'astuce se situe entre \\/ et la fin de ligne. Je vous conseille de lire les pages de manuel suivantes : *procmailex*, *procmailrc*, et *procmail*.

Par contre, il y a parfois des petits problèmes avec les gens qui changent constamment d'adresses, ou qui mettent des *anti-spams* (`NOSPAM` par exemple). Mais cela reste des incidents mineurs, et c'est facilement rattrapable.

Il y a ensuite les recettes au cas par cas, par exemple, si je veux que les mails de *Robert Dupond* aille dans $MAILDIR/dupond :

```
# ~/.procmailrc

PRIVDIR=$MAILDIR/private

:0:
* ^From:.*robert\.dupond@fai\.fr
$PRIVDIR/dupond
```

Si vous êtes sous *Gnus*, vous avez la possibilité de faire ce classement avec les mécanismes nnmail-split-*, couplés à votre carnet d'adresse *bbdb*[9]. Je n'ai pas encore essayé cette méthode, mais cela ne saurait tarder.

2.3 Enfin...

Ça y est, votre mail est enfin arrivé à destination, il ne me reste plus qu'à le lire. Et c'est bien là l'ennui, ne serait-ce pas le paradis si c'était également automatisé ? *:-)*

Après avoir longtemps utilisé *mutt*[10], j'utilise, depuis décembre 2001, le duo *emacs/Gnus*. Pourquoi avoir fait un retournement aussi brutal ? Car *mutt* est très bien tant que l'on reste dans la norme, mais l'on est vite limité, contrairement à *Gnus*. En effet *mutt* n'est pas totalement configurable, et on est quasiment obligé d'utiliser *vi[m]* (si vous utilisez emacs comme éditeur, vous feriez mieux de changer de MUA[11]). Il suffit de voir comment est organisé la liste des *mailboxes* dans *Gnus* et dans *mutt*, la différence est claire et nette.

Mais là n'est pas la question, nous n'allons pas *troller* pour finalement tomber d'accord : *Gnus rulez*...

Je vous laisse imaginer comment je lis vos mails, et comment j'y réponds, de toutes façons, çela n'intéresse personne...

[9] http ://bbdb.sourceforge.net/
[10] http ://www.mutt.org/
[11] MUA : Mail User Agent

Chapitre 3

Laissons mourrir vos mails en paix

Ça y est, j'ai lu votre mail, et j'y ai peut-être répondu, maintenant, il doit mourrir dignement. Je m'occupe des funérailles, soyez sans crainte.

3.1 Expiration des mails

3.1.1 Listes de diffusion

Gnus permet d'expirer les mails, ce qui me permet de ne pas financer l'industrie du disque dur. Ce processus peut-être réalisé de deux manières : les articles non-lus pendant une certaine période sont automatiquement marqués comme lus. Et l'autre manière est de supprimer les mails lus trop vieux. Je n'utilise que la deuxième méthode, je supprime les mails lus au bout de deux mois. Ce qui représente tout de même plusieurs dizaines de méga-octets.

3.1.2 Mails personnels

Les mails personnels ne sont pas touchés par le processus d'expiration : ils sont gardés à vie (enfin, tant que mon disque dur est vivant plutôt).

3.2 Indexation des mails

Si je conserve mes mails privés, ce n'est pas pour rien. Mais si je dois passer trois heures avant de trouver un certain message, ce n'est pas la peine. J'utilise donc un moteur de recherche, qui est conçu entre autre pour les mails, c'est *swish++*, malgré l'avertissement dans la *FAQ*[1] qui nous avertit que son utilisation dans une langue autre que l'anglais n'est pas garantie. Tout fonctionne pourtant à merveille. Donc ne vous tracassez pas : les développeurs doivent être très frileux, ou trop modestes pour avouer que leur programme est excellent. Pour installer le logiciel, vous pouvez utiliser le paquet *Debian* au nom original de *swish++*, ou bien vous pouvez récupérer les sources sur Internet (Freshmeat

[1] FAQ : Frequently Asked Question

est votre ami). Sur ma machine, la base de recherche est réactualisée tous les jours par *crontab*, le matin très tôt, à 6h42 exactement.

La ligne de commande utilisée est celle-ci :

```
nbareil@mouarf:~ % crontab -l
42  6 *    *   *  nice -n 20  index++ -i /home/nbareil/Gnus/index.mail -e 'mail:*.*' -e 'm
```

De cette façon, j'indexe tous mes répertoires `~/Gnus`, et `~/Mail`.

Lorsque je veux faire une recherche, je pourrai utiliser directement l'outil fournit avec *swish++* qui est *search++*, mais comme je préfère que tout soit intégré à *Gnus*, j'utilise un *backend* écrit par le célèbre *Kai Grobjohann* : c'est `nnir` qui est disponible sur ftp ://ls6-ftp.cs.uni-dortmund.de/pub/src/emacs. Dans votre fichier `~/.gnus`, il vous suffit de mettre cela pour l'activer :

```
;; swish++ : Search engine
(require 'nnir)  ; Dispo dans le package Debian gnus-bonus-el
(setq nnir-search-engine 'swish++)
(setq nnir-swish++-program "search++") ; the search executable in Debian
(setq nnir-swish++-index-file "/home/nbareil/Gnus/index.mail");
(setq nnir-swish++-configuration-file "/home/nbareil/Gnus/swish++.conf")
```

Il ne vous reste alors qu'à taper `G-G` dans le *Group summary*, et vous voilà dans une fenêtre de recherche.

3.3 Sauvegarde des mails

Il y a encore quelques mois, je sauvegardais l'intégralité de mes mails sur une disquette *Zip* de 250 Mo, j'utilisais *GNU tar*, et *GNU Bzip2*, la *tarball* résultante avait une taille supérieure à 30 Mo malgré le niveau élevé de compression utilisé.

Mais ce type de sauvegarde est inefficace, car lorsque l'on veut récupérér un seul fichier, nous sommes obligés de tout décompresser dans un répertoire temporaire, de copier le fichier cible, et enfin de tout supprimer. Cette manipulation occupe un espace disque conséquent, et utilise intensivement le processeur.

La solution est *dump* que j'utilise désormais : ce programme, d'origine *BSD*, connu depuis des siècles par les administrateurs Unix est très solide, et son format est robuste. Ces avantages principaux sont la possibilité de :

1. Faire des sauvegardes incrémentales, c'est à dire n'archiver que les fichiers modifiés depuis une certaine date. On peut alors, grace à ce système, faire ses *backups* tous les jours sans crainte pour son espace disque.
2. Pouvoir restaurer un unique fichier sans avoir besoin de tout décompresser, et d'une facon transparente.

Perl arrive encore à notre secours pour faciliter les opérations de sauvegardes : j'ai écrit un petit *front-end* à *dump* qui se charge de tout. Le script est disponible en annexe (ou sur mon site encore une fois).

Une des fonctionnalités future est le chiffrement des sauvegardes, par simple paranoïa (en utilisant GnuPG).

Chapitre 4

Conclusion

Voila, c'est terminé, vous avez pu voir le cycle complet d'un message électronique sur ma machine qui n'est pas de tout repos. Les ordinateurs chargés de cette tâche ne sont pas des bêtes de course, ce sont pour la plupart des *Pentium 133* au maximum. Le besoin en ressource n'est donc pas très important.

Si vous avez des questions, n'hésitez pas à m'envoyer un message aux adresses indiquées dans le chapitre I, je me ferai une joie d'y répondre, et puis c'est toujours agréable de recevoir des retours...Autrement, je vous conseille la lecture du forum *Usenet* `fr.comp.mail`, mais munissez vous d'un *score file* bien rempli, car les discussions autour de *Microsoft Windows* sont nombreuses.

Je remercie chaleureusement mes deux relectrices de charme : *Amal B.*, et *Lansciac* de *madchat.org*. C'est grâce à elles que ce document comporte le moins de fautes possibles. Tant qu'on est dans les remerciements, merci à : *Léonard*, *Romain Gardon*, *Trankillou*, *Ulysse*, *Wolf Cykl*, *Tobozo*, *Cybz*, *Christophe Casalegno*, *Com boy*, *Khalaan*, *Torkal*...

Bref, merci à tous ceux qui me supportent au quotidien.

Chapitre 5

Front-end de dump

```
#! /usr/bin/perl -w

use strict;
use Pod::Usage;
use Getopt::Long;

##############################################################
#  Ce script est auto-documenté, les pages de manuel sont   #
# disponibles en executant ce script avec le flag --man. Ou #
# en utilisant la commande pod2man (dispo sur toutes les    #
# machines avec /usr/bin/perl installé).                     #
##############################################################

# Valeur par défaut des options
my $verbose  = 0;
my $man      = 0;
my $help     = 0;

# On récupère nos options en ligne de commande
GetOptions('help!'          => \$help,
           'man!'           => \$man,
           'verbose+'       => \$verbose,
          ) or pod2usage(2);

pod2usage(1) if $help;
pod2usage(-exitstatus => 0, -verbose => 2) if $man;

# Partitions différentes
my %data = (
            # Une clé par partition
            'home' => {# Là où est la liste des fichiers à archiver
                       'file' => '/var/backups/dump/files_home',
                       # Le nom du fichier de backup final (sans le .bz2)
                        'backup'=> '/var/backups/dump/backup_home'
                      },
```

```
            'etc'  => { 'file' => '/var/backups/dump/files_etc',
                        'backup'=> '/var/backups/dump/backup_etc'
                      },
            'var'  => { 'file' => '/var/backups/dump/files_var',
                        'backup'=> '/var/backups/dump/backup_var'
                      }
          );

# Variables globales
my $DUMP = "/sbin/dump";
my $BZIP = "/usr/bin/bzip2";
my $OPTIONS_DUMP;
my $current;
my @files;

# main()
die "Le programme $DUMP n'existe pas !" unless -x $DUMP;

for my $key (keys %data) {
  print "\n\n[*] Partion en cours : $key\n";

  $current = $data{$key};

  unless (-f $current->{"file"}) {
    print STDERR "ERR: Le fichier de configuation $key n'existe pas !\n";
    print STDERR "ERR: Cette partition ne va donc pas être traité\n";
    next;
  }

  # On vide @files
  @files = ();

  # On liste tous les fichiers à archiver
  open FD, $current->{file}
    or die "open($current->{file}) : $!\n";

  while ( <FD> ) {
    chomp if defined;
    push @files, $_;
  }

  close FD
    or die "$!\n";

  $OPTIONS_DUMP  = "-0 -f $current->{backup} ";
  $OPTIONS_DUMP .= join " ", @files;

  print "$DUMP $OPTIONS_DUMP 2>&1\n" if $verbose;

  # On execute dump avec les bonnes options
```

```
  open dump_proc, "| $DUMP $OPTIONS_DUMP 2>&1"
    or die "Problème : $!\n";

  # FIXME : Le test de la valeur de retour n'indique pas la réussite
  # de la commande, juste que le fork s'est bien déroulé.
  # Donc à prendre avec des pincettes, et le code de test doit
  # vraiment être amélioré pour faire un test correct.

  # On affiche toute la sortie de dump
  print while <dump_proc>;

  close dump_proc
    or die "$!\n";


  # On compresse l'archive
  open bzip2_proc, "| $BZIP $current->{backup}"
    or die "open() $!\n";

  print while <bzip2_proc>;

  close bzip2_proc or die $!;

  # Faut tester l'intégrité de l'archive
  open test_proc, "| $BZIP -t $current->{backup}.bz2"
    or die "open() $!\n";

  print while <test_proc>;

  close test_proc or die $!
}

__END__

=head1 NAME

make-backup - Front-end pour /sbin/dump

=head1 SYNOPSIS

./make-backup [options]

Options:
  --help            Affiche les options disponibles,
  --man             Affiche la page de manuel,
  --verbose         Mode verbeux (flag cumulatif),

=head1 OPTIONS

=over 8
```

```
=item B<--help>

Rappel de toutes les options disponibles, variables d'environnement,
syntaxe de la ligne de commande.

=item B<--man>

Affiche la page de manuel au format nroff.

=item B<--verbose>

Affiche tous les messages de log, même ceux qui sont insignifiants.

=back

=head1 DESCRIPTION

On peut considérer ce script comme un front-end à dump, en effet, vous
n'avez que 2 choses à faire : Créér des fichiers de configuration
indiquant les fichiers (ou répertoires) à être sauvegardé, et
indiquer l'endroit de ces fichier de conf dans ce script.  Dump ne
pouvant pas faire des sauvegardes entre plusieurs partitions/disques
dur, vous êtes obligé de faire un fichier pour chaque partition.  Par
exemple, chez moi, /etc, /var, et /home sont des partitions
différentes, donc pour sauvegarder ces partitions, il faut créér 3
fichiers différents.  Dans ces fichiers, vous indiquez un
fichier/répertoire par ligne.  Et vous n'avez qu'à indiquer l'endroit
de ce fichier au script en éditant la ligne 130.
Chez moi :

mouarf:~# ls /var/backups/dump/
files_etc  files_home  files_var
mouarf:~# cat /var/backups/dump/files_var
/var/www
/var/cvs

Et dans les script, je mets :
my %data = (
            'var'  => { 'file' => '/var/backups/dump/files_var',
                        'backup'=> '/var/backups/dump/backup_var'
                      }
          );

Et après le passage du script, ma backup (compressé par bzip2) sera le
fichier /var/backups/dump/backup_var.bz2

=head1 DEPENDANCES

Votre système doit avoir B</sbin/dump> d'intallé, ainsi que B</usr/bin/bzip2>.
```

Dump peut être téléchargé ici : B<http://dump.sourceforge.net/>.

=head1 LICENCE

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2, or any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA
02111-1307, USA.

=head1 DATE

10/09/2002

=head1 AUTEUR

Nicolas Bareil (offset @ madchat.org)
http://www.matoufou.org/linux/

=cut