Installation et Configuration de Procmail

Procmail est un utilitaire très puissant et vraiment pratique pour gérer les mails reçus. Il permet par exemple de filtrer des mails, de faire des redirections en fonction de l'envoyeur, du sujet, de la taille du mail,etc ..

Son utilisation et sa configuration sont assez simples. La configuration se résume par exemple à l'édition d'un seul et unique fichier.

1. Installation

Quelque soit votre distribution, elle est nécesssairement livrée avec un paquetage Procmail. Si ce n'est pas le cas, c'est peut-être le moment d'en changer.

- **Si vous êtes utilisateur de debian :**
  apt-get install procmail

- **Si vous êtes utilisateur de Mandriva :**
  urpmi procmail

- **Si vous utilisez Fedora,** regardez dans vos cdroms et installer procmail ainsi :
  rpm -Uvh procmail-xxx.rpm

- **Si vous utilisez Slackware,** regardez dans votre cdrom (répertoire slackware/n/) et installer procmail ainsi :
  installpkg procmail-xxxx.tgz

Pour ceux qui souhaitent savoir comment compiler Procmail ou ceux qui veulent l'installer depuis les sources, je détaille l'installation à partir des sources. Pour ce faire, téléchargez les sources de Procmail ICI et installez-le comme suite :
$ tar xzf procmail-x-xx.tar.gz
$ cd procmail-x-xx
$ su
# make install
# make install-suid

Et voilà, Procmail est installé.

2. Configuration de Procmail

La configuration de 'Procmail' est vraiment très simple et se résume à l'édition d'un fichier. Le nom de ce fichier importe peu car il sera donné en argument lorsque l'on lancera 'procmail', néanmoins, par défaut, si aucun fichier de configuration n'est donné en argument à procmail, ce dernier va lire le

fichier $HOME/.procmailrc ou en deuxième recours, le fichier /etc/procmailrc. Le fichier de configuration est composé de deux parties.

Partie I :

Contient les variables qui seront utiles aux différents scripts et programmes lancés.

```
# mettre /bin/sh surtout si vous utilisez tcsh !

SHELL=/bin/sh

# répertoire où seront stockés les mails

MAILDIR=/home/moi/Mail

# chemin d'accès aux exécutables ; en mettre le minimum, pour n'accéder qu'aux

# programmes indiqués dans le fichier de configuration

PATH=/bin:/usr/bin:/usr/local/bin

# si procmail n'arrive pas à délivrer le courrier, cette boîte sera utilisée.

ORGMAIL=$MAILDIR/secours

# boîte de réception par défaut

DEFAULT=/var/mail/default

# Fichier de log de procmail

LOGFILE=$MAILDIR/.procmail.log

# Fichier de configuration a inclure

INCLUDERC=/etc/procmailrc-config-general
```

Partie II :

Contient une série de blocs dont la syntaxe éxacte sera la suivante :

```
:0 [flags] [ : [locallockfile] ]

        < zéro ou plusieurs conditions (mais seulement une seule par ligne) > < éxactement une ligne d'action >
```

Les conditions :

- elles commencent toujours par le caractère '*'.
- tout ce qui se trouve après le caractère '*' est analysé, sans aucune modification, par "egrep"(interne à procmail). Il faut donc que ce soit une expression régulière compatible pour "egrep"(man egrep)

La ligne d'action :

- sauvegarder le mail: on écrit tout simplement le chemin absolu ou relatif (dans ce cas, le répertoire d'origine est $MAILDIR) du répertoire ou l'on veut stocker une copie du mail.
- envoyer le mail vers une autre adresse mail : on utilise le point d'exclamation "!" (Ex: ! nom@domaine.fr)
- si la, ou les conditions sont valide, on peut envoyer le mail vers un groupe de blocs, en regroupant le tout par des accolades "{}".
- commande : elle doit commencer par un pipe (le caractère '|'). IMPORTANT : il faut bien noter que cette ligne reçoit le "flux d'entrée", c'est à dire le mail dans le cas présent. Si l'action ne renvoit aucun "flux de sortie", cela revient à renvoyer au prochain bloc un mail vide. C'est pourquoi, dans cette situation, il faut utiliser le flag 'c' qui créera une copie conforme du mail et l'enverra au prochain bloc.

Liste des flags :

- H Recherche de motif dans l'entête du mail(défaut).
- B Recherche de motif dans le corps du mail.
- D Distinguer les lettres majuscules, des minuscules (contrairement à la règle par défaut, qui ne fait pas la distinction).
- A Le bloc serat exécuté si, et seulement si, la condition du précédent bloc(sans flag 'A' ou 'a') concorde aussi. Cela permet d'enchaîner des actions qui dépendent d'une même condition.
- a Idem que le flag 'A', avec la condition supplémentaire que le précédent bloc s'est terminé avec succès.
- E Ce bloc serat exécuté si, et seulement si, le précédent bloc n'a pas été exécuté. L'exécution de ce bloc annule les blocs suivant qui contiennent le flag 'E', permettant ainsi d'avoir le rôle d'un 'else if'.
- e Ce bloc serat exécuté si, et seulement si, le précédent bloc fût un échec.
- h Renvoie le contenu de l'entête du mail vers le 'pipe',le fichier ou le mail d'arrivé(défaut).
- b Renvoie le corps du mail vers le 'pipe',le fichier ou le mail d'arrivé(défaut).
- f Considère le 'pipe' comme un filtre.
- c Génère une copie conforme du mail.
- w Attend que le filtre ou le programme se finisse et vérifie son 'exitcode' (normalement ignoré); si le filtre n'est pas un succès, cela signifie que le texte ne doit pas avoir été filtré.
- W Idem que pour 'w', mais supprime les méssages d'érreurs qui pourraient survenir.
- i Ignore les possibles érreurs d'écriture dans ce bloc.
- r Raw mode. Ne s'assure pas que le mail se termine bien par une ligne vide, et recopie ce dernier tel quel.

Voici quelques exemples de "Paritie II" :

- **Exemple 1 :** Nous voulons que tout mail contenant la chaîne de caractères 'trustonme' dans son sujet soient mis dans le fichier trust

```
:0:

* ^Subject.*trustonme

trust
```

- **Exemple 2 :** Nous voulons que tous les mails en provenance de Linus (Torvalds) soient mis dans le fichier linus

```
:0c:
```

```
* ^From.*linus

linus
```

- **Exemple 3 :** Ce qui suit est un filtre pour mailing-list. Nous souhaitons que tous les messages en provenance de trustforum, contenant le mot bug dans leurs corps (B comme body) vont dans le fichier bug, les autres vont dans funct.

```
:0

* ^To:.*trustforum

{

:0cB:

* bug

bug

:0cB:

* ! bug

funct

}
```

- **Exemple 4 :** On cherche ici à filtrer(f) le spam en utilisant 'spamassassin'(| spamc). Puis, quand ce dernier a terminé(w), on recherche dans l'en-tête du mail si le motif "X-Spam-Status: Yes" est présent. Si c'est le cas, on met le mail dans le dossier $MAILDIR/spam/, autrement le mail va dans $MAILDIR/new :

```
:0fw

| spamc



:0

* ^X-Spam-Status: Yes

spam/



:0

new/
```

- **Exemple 5 :** Dans cet exemple, on vérifie que le mail n'a pas été reconnu comme spam, puis, si ce n'est effectivement pas du spam, on lance deux autres blocs, regroupés grâce aux accolades. Le premier de ces blocs scanne le mail avec l'antivirus "clamav", et le second envoie le mail à "toto@mon_domaine.com" :

```
:0
```

```
* !^X-Spam-Status: Yes

{

:0fw

| /usr/bin/clamfilter.pl


:0

! toto@mon_domaine.com

}
```

Vous trouverez beaucoup plus d'exemples intéressants en faisant un 'man procmailex' ...

3. Conclusion

Procmail est désormais installé et configuré. Néanmoins, tel quel, il ne vous serat d'aucune utilité. En effet, il vous faut désormais le coupler avec le Serveur SMTP que vous utilisé. A titre d'information, sachez que le langage de programmation le plus adapté à la recherche de motif est, à mon avis, le langage PERL. Si vous désirez vous créer votre propre filtre 'fait maison', c'est de ce côté là qu'il faut chercher.