

IMAP vs POP

POP3

```
$ telnet mailhost 110
+OK POP3 server ready

USER pdupont
+OK Name is a valid mailbox
PASS tululu
+OK Maildrop locked and ready

LIST
+OK scan listing follows
1 169
2 811
3 813
.

RETR 1
+OK Message follows
Return-Path: <David.Olivier@univ-lyon2.fr>
From: David.Olivier@univ-lyon2.fr
To: David.Olivier@etu.univ-lyon2.fr
Subject: essai chien chapeau

essai chien chapeau
.

DELE 1
+OK message deleted

QUIT
+OK
Connection closed by foreign host.
$ date
L'heure du café
$
```

IMAP4

```
$ telnet mailhost 143
* OK IMAP4 server ready
. LOGIN pdupont tululu
. OK User logged in
. SELECT INBOX
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
* OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted
* 3 EXISTS
* 3 RECENT
* OK [UNSEEN 1]
* OK [UIDVALIDITY 901006906]
. OK [READ-WRITE] Completed

. UID FETCH 1:* RFC822.SIZE
* 1 FETCH (UID 1425 RFC822.SIZE 169)
* 2 FETCH (UID 1426 RFC822.SIZE 811)
* 3 FETCH (UID 1427 RFC822.SIZE 813)
. OK Completed

. UID FETCH 1425 BODY[]
* 1 FETCH (FLAGS (\Recent \Seen) UID 1425 BODY[] {169})
Return-Path: <David.Olivier@univ-lyon2.fr>
From: David.Olivier@univ-lyon2.fr
To: David.Olivier@etu.univ-lyon2.fr
Subject: essai chien chapeau

essai chien chapeau
)
. OK Completed

. UID STORE 1425 +FLAGS (\Deleted)
* 1 FETCH (FLAGS (\Recent \Deleted \Seen) UID 1425)
. OK Completed

. EXPUNGE
* 1 EXPUNGE
* 2 EXISTS
* 2 RECENT
. OK Completed
. LOGOUT
* BYE LOGOUT received
. OK Completed
Connection closed by foreign host.
$ date
L'heure du café
$
```

Sommaire

- [Généralités](#)
 - [Commandes unix : regarder pas toucher](#)
 - [Format des commandes](#)
 - [Format des réponses du serveur](#)
 - [Encapsulation des données](#)
 - [Login et logout](#)
 - [Organisation des boîtes](#)
 - [Les droits sur les boîtes](#)
 - [Les quotas](#)
 - [Les numéros de séquence des messages et les UID](#)
 - [Les flags des messages](#)
- [Commandes de manipulation des boîtes](#)
 - [Listage de boîtes](#) (LIST)
 - [Création d'une boîte](#) (CREATE)
 - [Suppression d'une boîte](#) (DELETE)
 - [Modification du nom d'une boîte](#) (RENAME)
 - [Listage/modification des droits sur une boîte](#) (GETACL / SETACL)
 - [Les commandes relatives aux quotas](#) (SETQUOTA / GETQUOTA / GETQUOTAROOT)
 - [Insertion d'un message dans une boîte](#) (APPEND)
 - [Sélection d'une boîte](#) (SELECT / EXAMINE / CLOSE)
- [Commandes de manipulation des messages de la boîte sélectionnée](#)
 - [Listage de messages](#) (SEARCH)
 - [Lecture de tout ou partie d'un message](#) (FETCH)
 - [Recopie d'un message vers une autre boîte](#) (COPY)
 - [Modification des flags d'un message](#) (STORE)
 - [Purge de la boîte](#) (EXPUNGE)

Généralités

Cette page se veut une introduction rapide et pratique à IMAP4, et un répertoire/aide-mémoire de commandes magiques.

Les informations données sont pour la plupart relatives à IMAP en général ; certaines, cependant, sans que ce ne soit toujours précisé, sont plutôt spécifiques au serveur IMAP utilisé à Lyon 2 (serveur Cyrus v1.5.14), voire à la configuration précise sur `louis` ou `etienne`.

Ces informations ne sont pas complètes, loin de là. Pour plus de précision et d'exhaustivité, il faut voir les RFC. Les principales qui peuvent nous intéresser sont :

[RFC 2060](#)

Le protocole IMAP4rev1

[RFC 2086](#)

Les ACL (droits d'accès sur les boîtes ; c'est une extension du protocole IMAP4)

[RFC 2087](#)

Les quota (limitation de taille des boîtes ; c'est une extension du protocole IMAP4)

L'implémentation du serveur Cyrus est documentée dans leur page [«Overview and Concepts»](#).

Comme documentation plus introductive aux différences/ressemblances entre POP3 et IMAP4, on pourra consulter [imap.vs.pop](#), version française abrégée de [imap.vs.pop](#) de Terry Gray (Un. of Washington).

Commandes unix : regarder pas toucher

Pratiquement toute l'administration des boîtes doit se faire à travers le protocole IMAP lui-même (ou par POP3, éventuellement), sans jamais toucher directement aux fichiers par les commandes unix. Sinon, on risque fort d'introduire des incohérences.

```
$ telnet louis imap # imap = 143
Connected to louis.univ-lyon2.fr.
Escape character is '^]'.
* OK louis Cyrus IMAP4 v1.5.14 server ready
```

Format des commandes

Toutes les commandes sont à préfixer par un *tag*, reproduit dans la réponse du serveur :

```
gruk1 LOGIN pdupont tululu
gruk1 OK User logged in
gruk2 SELECT INBOX
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
(...)
gruk2 OK [READ-WRITE] Completed
```

Ce tag, qu'on doit choisir en principe à chaque fois différent, permet la synchronisation dans le cas d'un client qui tape très très vite et envoie plusieurs commandes sans attendre la réponse des précédentes. Si vous ne tapez pas très très vite, un simple point suffira :

```
. LOGIN pdupont tululu
. OK User logged in
. SELECT INBOX
(...)
. OK [READ-WRITE] Completed
```

Le nom des commandes peut aussi être tapé en minuscules, c'est plus facile. Je les mets ici en majuscules, c'est plus clair.

Les paramètres sont à séparer par un espace et un seul. Quand eux-mêmes contiennent des espaces, il est possible d'utiliser des doubles cotes :

```
. SELECT INBOX.Messages inutiles voire nuisibles
. BAD Unexpected extra arguments to Select
. SELECT "INBOX.Messages inutiles voire nuisibles"
(...)
. OK [READ-WRITE] Completed
```

Format des réponses du serveur

Il y a deux types de réponses : *tagged* et *untagged*.

À chaque commande du client correspond une et une seule réponse *tagged*, qui indique la fin du traitement de la requête. La réponse *tagged* est comme son nom le suggère préfixée du même *tag* que la commande correspondante :

```
monPremierTag LOGIN pdupont tululu
monPremierTag OK User logged in
```

Après le *tag* vient le code de complétion, qui peut être : OK (tout va bien), NO (pas bon) ou BAD (pas compris).

Les réponses *untagged*, quant à elles, commencent par un '*' et fournissent les informations requises par la commande et/ou diverses informations que le serveur estime le client en devoir de connaître :

```
monDeuxiemeTag SELECT INBOX
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
* OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen \*)] * 3
EXISTS
* 3 RECENT
* OK [UNSEEN 1] * OK [UIDVALIDITY 901006906] monDeuxiemeTag OK [READ-WRITE]
Completed
```

Le serveur peut aussi à tout moment envoyer des informations non sollicitées, sous forme *untagged* (préfixées par un '*'). De fait, l'interprétation d'une ligne *untagged* est le plus souvent indépendante de la commande qui l'a provoquée.

Encapsulation des données

En général, chaque commande, et chaque réponse (*tagged* ou *untagged*), correspond à une ligne et une seule, terminée par CRLF. Mais à l'intérieur de cette «ligne» il peut en fait y avoir des paramètres ou données binaires ou de n'importe quel format ; celles-ci peuvent être, et parfois doivent être, «encapsulées» par la notation '{n}CRLF', *n* indiquant la longueur des données.

Ainsi, au lieu de :

```
. LOGIN pdupont tululu
```

on peut écrire, en encapsulant les 7 octets du paramètre pdupont :

```
. LOGIN {7}CRLFpdupont tululu
```

c'est-à-dire :

```
. LOGIN {7}
pdupont tululu
```

Le dialogue réel se présente ainsi :

```
. LOGIN {7}
+ go ahead
pdupont tululu
. OK User logged in
```

car, dans le cas d'une commande, le serveur envoie la ligne + go ahead pour se déclarer disposé à recevoir les données.

Dans l'exemple ci-dessus, l'encapsulation ne sert évidemment à rien, mais c'est par ce moyen qu'est échangé, par exemple, le contenu d'un mail. L'exemple donné en pré-générique :

```
. UID FETCH 1425 BODY[]
* 1 FETCH (FLAGS (\Recent \Seen) UID 1425 BODY[] {169}
Return-Path: <David.Olivier@univ-lyon2.fr>
From: David.Olivier@univ-lyon2.fr
To: David.Olivier@etu.univ-lyon2.fr
Subject: essai chien chapeau
essai chien chapeau
)
. OK Completed
```

s'analyse donc comme suit :

```
. UID FETCH 1425 BODY[]
* 1 FETCH (FLAGS (\Recent \Seen) UID 1425 BODY[] {169}CRLF[169 octets])
. OK Completed
```

On voit que syntactiquement les données encapsulées se comportent comme un atome. La parenthèse fermante qui termine la «ligne» correspond à la parenthèse ouvrante qui suit FETCH.

Login et logout

Le login est la première commande à taper. `pdupont` peut faire :

```
. LOGIN pdupont tululu
. OK User logged in
```

Le login d'administration `cyrus` (déclaré dans `/etc/imapd.conf`) permet d'avoir accès à toutes les boîtes. `cyrus` n'a pas automatiquement tous les droits sur toutes les boîtes, mais seulement *le droit de se donner tous les droits* (droit implicite 'a' (administration)).

Les logins/mots de passe sont ceux reconnus par le système sur lequel tourne le serveur. Un utilisateur peut très bien se logger même s'il ne possède pas de boîte :

```
. LOGIN pdupont tululu
. OK User logged in
. SELECT user.pdupont
. NO Mailbox does not exist
```

La dernière commande à faire est le logout :

```
. LOGOUT
* BYE LOGOUT received
. OK Completed
```

Si l'utilisateur s'endort sur son clavier plus d'une demi-heure le serveur coupe la connexion. Mais si dans son sommeil il tape de temps en temps

```
. NOOP
```

le serveur le croit encore éveillé.

Organisation des boîtes

Dans le fichier de configuration `/etc/imapd.conf` on trouve deux lignes du genre :

```
defaultpartition: turlu
partition-turlu: /var/mail
```

Les boîtes sont alors dans `/var/mail`.

Elles sont organisées en arborescence. La boîte principale de `pdupont` est dans `/var/mail/user/pdupont` ; par le protocole IMAP, elle apparaît comme `user.pdupont`. Ainsi :

Hiérarchie unix Hiérarchie IMAP

```
/var/mail/a/b/c a.b.c
```

L'utilisation du point ('.') comme séparateur hiérarchique ne découle pas du protocole IMAP, mais de son implémentation dans le serveur Cyrus. C'est pourquoi les clients IMAP demandent souvent à l'utilisateur de spécifier ce séparateur dans les préférences.

Les noms de dossiers doivent être de l'ascii pur. Par convention, les caractères non ascii (accents...) doivent être codés «mUTF7» par le client. Ce n'est pas grave, simplement il ne faut pas s'étonner de rencontrer des noms comme «INBOX.MessagesEnvoy&AOk-s».

Les dossiers de la forme `user.login` ont quelques particularités :

- Si `login` est un vrai login et qu'on se connecte sous ce login, le nom `INBOX` devient synonyme de `user.login`.
- Le courrier reçu par `sendmail` à destination du login unix `login` est déposé (par l'intermédiaire de `deliver`) dans le `INBOX` de `login`, c'est-à-dire dans `user.login`.
- Si on choisit de snobber IMAP4 et d'opérer par POP3, on n'accédera qu'au `user.login`.
- L'utilisateur `login` a un droit implicite d'administration sur `user.login` et sur toute sous-boîte de `user.login`.
- La commande `RENAME` a un comportement particulier sur ces boîtes. Voir [ci-dessous](#).

Les messages sont stockés à raison d'un par fichier. Le nom de ceux-ci sont du genre «23.» :

```
$ ls /var/mail/user/pdupont
23. cyrus.header
24. cyrus.index
Corbeille cyrus.seen
cyrus.cache
```

Corbeille est ici la sous-boîte `user.pdupont.Corbeille` de `user.pdupont`.

Le numéro affecté à chaque message est en fait son «UID» (*Unique Identifier*). Sa valeur croît de 1 à chaque message successif dans une boîte donnée, de manière à identifier les messages de façon univoque.

La hiérarchie IMAP a ceci de particulier qu'une boîte `a.b.c` peut exister sans que la boîte `a.b` n'existe. Le fait que `user.pdupont` existe en tant que boîte IMAP est attesté par la présence des fichiers `cyrus.cache`, etc.

Les droits sur les boîtes

Il s'agit d'une extension au protocole IMAP : [RFC 2086](#).

Chaque boîte a une ACL (*Access Control List*), liste de couples login/droits :

```
. GETACL Public.Mulberry
* ACL Public.Mulberry cyrus lrswipcda anyone lr pdupont a
. OK Completed
```

ce qui veut dire que :

L'identifiant a les droits

```
cyrus lrswipcda
anyone lr
pdupont a
```

L'identifiant `anyone` veut dire n'importe quel login. Les identifiants utilisés ici peuvent être des logins, mais peuvent aussi correspondre à des groupes unix, sous la forme, par exemple, `<<group:staff>>`.

Il y a 9 droits possibles : `lrswipcda`.

Droit Description

- l Rend la boîte visible (listable).
- r Permet de voir et de lire le contenu de la boîte (les messages).
- s Les flags `\Seen` et `\Recent` sont modifiables.
- w Permet la modification des flags (autres que `\Seen` et `\Deleted`).
- i Permet l'insertion de nouveaux messages.
- p Autre droit lié à l'insertion, de façon obscure.
- c Permet de créer des sous-boîtes.
- d Permet de supprimer des messages ; permet de supprimer la boîte.
- a Permet d'administrer (modifier) les droits sur la boîte, donc indirectement de tout faire.

Outre les droits résultant des ACL, il existe des droits *implicites*. `cyrus` a un droit implicite 'a' (administration) sur toute boîte. Tout utilisateur a ce même droit 'a' sur ses propres boîtes (`user.login...`).

Ainsi, `cyrus` a toujours, au moins indirectement, tous les droits sur toutes les boîtes. Encore faut-il qu'il rende ces droits explicites, au besoin :

```
. LOGIN cyrus xxxxx
. OK User logged in
. GETACL Shabada
* ACL Shabada pdupont lrswipcda
. OK Completed
. DELETE Shabada
. NO Permission denied
. SETACL Shabada cyrus d
. OK Completed
. GETACL Shabada
* ACL Shabada pdupont lrswipcda cyrus d
. OK Completed
. DELETE Shabada
. OK Completed
```

Il en va de même, en principe, pour les utilisateurs relativement à leurs propres boîtes ; sauf que lors de la création d'une boîte `user.login`, tous les droits sont donnés explicitement à `login`.

En effet, lorsqu'une boîte est créée :

1. s'il s'agit d'une boîte `user.login`, son ACL est `login lrswipcda` ;
2. sinon, si la boîte a un parent dans la hiérarchie (immédiat ou non), son ACL est celui de son parent ;
3. sinon, son ACL est tel qu'indiqué dans `/etc/imapd.conf` :
`defaultacl: cyrus lrswipcda`

L'héritage décrit dans le point 2. est statique : si on modifie ultérieurement l'ACL de `a.b`, l'ACL de `a.b.c` n'est pas automatiquement modifié.

Les quotas

Les quotas sont des limitations imposées sur des groupes de boîtes. Le serveur Cyrus implémente le quota «STORAGE», relatif au volume totale des messages. Les quotas sont une extension à IMAP4, définie par la [RFC 2087](#).

Les volumes pris en compte par les commandes quota «STORAGE» sont la taille des messages en octets. La place réellement occupée sur le disque est donc supérieure, en raison de l'allocation des fichiers par blocs et de la présence de divers fichiers annexes (index, etc.).

Même quand on n'entend pas limiter le volume d'une boîte, il peut être utile de la soumettre à un quota, car cela permet de connaître à chaque instant le volume occupé.

Les quotas se basent sur la création de *quota roots* affectées à un point donné de la hiérarchie. Une *quota root* de 500 ko affectée au point `Shabada.Wawa`, par exemple, limitera à 500 ko le volume total de toute boîte dans la sous-hiérarchie `Shabada.Wawa[.*]`. Ceci sauf s'il existe au sein de cette sous-hiérarchie une autre *quota root*. Chaque boîte est en effet soumise à au plus une seule *quota root*, celle qui est la plus proche en remontant la hiérarchie.

Si ceci n'est pas clair, vous n'avez qu'à aller voir sur la page de présentation du serveur, [«Overview and Concepts»](#). Ils arrivent à expliquer mieux que moi. Mais c'est en anglais. Ils donnent comme exemple :

On a les boîtes suivantes :

```
user.bovik
user.bovik.list.imap
user.bovik.list.info-cyrus
user.bovik.saved
user.bovik.todo
```

ainsi que les *quota roots* :

```
user.bovik user.bovik.list
user.bovik.saved
```

Alors, la *quota root* `user.bovik` gouverne les boîtes `user.bovik` et `user.bovik.todo`. La *quota root* `user.bovik.list` gouverne les boîtes `user.bovik.list.imap` et `user.bovik.list.info-cyrus`. La *quota root* `user.bovik.saved` gouverne la boîte `user.bovik.saved`.

Le quota fixé interdit normalement d'insérer un message dans une des boîtes qui en dépendent si cette insertion ferait dépasser le volume autorisé. Cependant, lorsqu'il s'agit d'un message arrivant par `sendmail` (par l'intermédiaire de `deliver`), le comportement est un peu différent. Les messages sont acceptés jusqu'à et y inclus celui qui fait dépasser le volume autorisé. Ce sont les suivants qui sont refusés. Si par exemple on impose à un utilisateur un quota de 1024 ko, qu'il en utilise 50, et qu'arrive un message de 5000 ko, celui-ci sera accepté. Le message suivant sera rejeté, même s'il ne fait que 400 octets. Le monde est souvent injuste.

Les numéros de séquence des messages et les UID

Chaque message d'une boîte donnée est référencé par deux numéros : son numéro de séquence et son UID (*Unique Identifier*).

Les numéros de séquence sont simplement des numéros de 1 à n , nombre de messages dans la boîte. Le numéro de séquence d'un message donné peut changer, même au cours de la session (si on efface un message, les numéros des messages suivants sont décrémentés).

Les UID, par contre, désignent les messages de façon permanente. Ils sont attribués dans l'ordre d'arrivée des messages dans la boîte, et jamais réutilisés (souvenons-nous de cette précieuse qualité des nombres d'exister en quantité infinie).

Des messages dans des boîtes différentes peuvent par contre avoir le même UID.

Les flags des messages

À chaque message sont associés les flags suivants :

```
\Seen      Message déjà lu
\Answered  Message auquel il a été répondu
\Flagged   Message important
\Deleted   Message marqué pour effacement
\Draft     Brouillon de message
\Recent    Message tout nouveau
```

Pour l'administration importe surtout le flag `\Deleted`. Pour supprimer un message, on commence par le marquer `\Deleted` ; puis on purge la boîte, ce qui en supprime définitivement à jamais tout message `\Deleted`.

Les commandes de manipulation des boîtes

Listage de boîtes

Pour lister toutes les boîtes visibles du login sous lequel on s'est loggué :

```
. LIST "" *
* LIST () "." "INBOX"
* LIST () "." "INBOX.Corbeille"
* LIST () "." "INBOX.Trucs"
* LIST () "." "INBOX.Trucs.Machins"
* LIST () "." "Jardin.Secret.Suspendu"
. OK Completed
```

Les boîtes listées sont celles qui sont visibles de l'utilisateur (droit 'l'). Attention : si on s'est loggué sous `cyrus`, la liste risque d'être longue.

Liste partielle :

```
. LIST "" INBOX.%
* LIST () "." "INBOX.Corbeille"
* LIST () "." "INBOX.Trucs"
. OK Completed
```

Le caractère '%' correspond à toute chaîne sans séparateur hiérarchique ('.'). Le caractère '*' par contre correspond à n'importe quelle chaîne :

```
. LIST "" INBOX.*
* LIST () "." "INBOX.Corbeille"
* LIST () "." "INBOX.Trucs"
* LIST () "." "INBOX.Trucs.Machins"
. OK Completed
```

Création d'une boîte (boîte principale ou dossier)

Exemple : INBOX de `pdupont` :

```
. CREATE user.pdupont
. OK Completed
```

Boîte quelconque :

```
. CREATE Jardin.Secret.Suspendu
. OK Completed
```

Rappel : La création de Jardin.Secret.Suspendu n'implique pas l'existence des boîtes Jardin.Secret ni de Jardin.

Sous-boîte à pdupont :

```
. CREATE user.pdupont.Claude
. OK Completed
```

ou, si on s'est loggué sous pdupont :

```
. CREATE INBOX.Claude
. OK Completed
```

Supression d'une boîte

```
. DELETE user.pdupont.Claude
. OK Completed
```

La suppression d'une boîte n'implique pas la suppression des éventuelles boîtes de niveau inférieur. Par exemple, s'il existe les deux boîtes :

```
Jardin.Secret
Jardin.Secret.Suspendu
```

et qu'on supprime la première, la deuxième continue à exister.

Exception (serveur Cyrus) : Cette règle ne vaut pas pour les boîtes de la forme `user.login`. Ainsi, `DELETE user.pdupont` entraîne la suppression par le serveur de toute boîte `user.pdupont.*`. Il s'agit ici d'une entorse au protocole officiel.

Pour supprimer une boîte, il faut avoir le droit 'd' sur la boîte. Si on ne l'a pas, mais qu'on a le droit de se le donner (login `cyrus`, par exemple), faire :

```
. SETACL user.pdupont cyrus +d
. OK Completed
```

Modification du nom d'une boîte

Le schéma est :

```
. RENAME ancien_nom nouveau_nom
```

Exemple :

```
. RENAME user.pdupont.Claude user.pdupont.Dominique
. OK Completed
```

Si le nouveau nom indiqué existe déjà, l'opération n'est pas accomplie (réponse NO).

Renommage d'une boîte d'utilisateur (type `user.login`) : le comportement du server est particulier et embêtant. Pour renommer `user.pdupont` en `user.dupontp`, utiliser la séquence suivante :

```

. LOGIN cyrus xxxxxxxx
. OK User logged in
. CREATE user.dupontp # Création de la nouvelle boîte
. OK Completed
. SETACL user.dupontp cyrus lrswipcda
. OK Completed
. SETACL user.pdupont cyrus lrswipcda
. OK Completed
. SELECT user.pdupont
...
* 3 EXISTS
...
. OK [READ-WRITE] Completed
. COPY 1:* user.dupontp # Recopie des messages de l'ancienne boîte vers la
nouvelle
. OK ...
. LIST "" user.pdupont.* # Listage des sous-boîtes
* LIST () "." "user.pdupont.Corbeille"
* LIST () "." "user.pdupont.MessagesEnvoy&AOk-s"
. OK Completed
. RENAME "user.pdupont.Corbeille" "user.dupontp.Corbeille" # Renommage des
sous-boîtes
. OK Completed
. RENAME "user.pdupont.MessagesEnvoy&AOk-s"
"user.dupontp.MessagesEnvoy&AOk-s"
. OK Completed
# Vérifications :
. LIST "" user.pdupont.* # Il n'y a plus de sous-boîtes dans l'ancienne
boîte
. OK Completed
. LIST "" user.dupontp.* # On retrouve bien nos sous-boîtes dans la
nouvelle boîte
* LIST () "." "user.dupontp.Corbeille"
* LIST () "." "user.dupontp.MessagesEnvoy&AOk-s"
. OK Completed
. SELECT user.dupontp # Les messages de la boîte principale ont bien été
recopiés
...
* 3 EXISTS
...
. OK [READ-WRITE] Completed
# Et pour finir :
. DELETE user.pdupont
. OK Completed

```

Si certains RENAME échouent, il peut être nécessaire d'effectuer des SETACL *boîte cyrus lrswipcda* supplémentaires.

Bug du serveur IMAP Cyrus v1.5.14 :

Selon les spécifications, s'il y a des boîtes de niveau hiérarchique inférieur, elles sont renommées elles aussi. Ainsi, dans cet exemple, user.pdupont.Claude.BonnesBlagues doit devenir user.pdupont.Dominique.BonnesBlagues.

Mais la version actuelle du serveur ne le fait pas. Ils entendent corriger ça dans une version ultérieure.

Listage / modification des droits sur une boîte

Listage :

```
. GETACL Public.Mulberry
* ACL Public.Mulberry cyrus lrswipcda anyone lr pdupont a -claude lr
. OK Completed
```

La réponse donnée est une liste, sans ordre particulier (c'est en fait l'ordre de sa création), de couples login/droits. Le login peut être précédé du signe '-', qui veut dire que les droits en question lui sont enlevés. Ainsi, ici, tout le monde a le droit de lister et d'examiner la boîte Public.Mulberry (anyone lr), sauf l'utilisateur claude (-claude lr).

Modification :

```
. SETACL Public.Mulberry pdupont lrd
. OK Completed
. GETACL Public.Mulberry
* ACL Public.Mulberry cyrus lrswipcda anyone lr pdupont lrd -claude lr
. OK Completed
```

L'ACL pour pdupont devient exactement 'lrd'. pdupont a alors perdu les éventuels autres droits qu'il avait, c'est-à-dire dans le cas présent le droit 'a'.

Si ce qu'on veut est en fait *ajouter* des droits à pdupont, faire :

```
. SETACL Public.Mulberry pdupont +lrd
. OK Completed
. GETACL Public.Mulberry
* ACL Public.Mulberry cyrus lrswipcda anyone lr pdupont lrda -claude lr
. OK Completed
```

Si on veut enlever des droits :

```
. SETACL Public.Mulberry pdupont -r
. OK Completed
. GETACL Public.Mulberry
* ACL Public.Mulberry cyrus lrswipcda anyone lr pdupont lda -claude lr
. OK Completed
```

Commandes relatives aux quotas

Pour SETQUOTA et GETQUOTA, il semble qu'il faille être login cyrus pour que ça marche.

Création d'une *quota root* :

```
. SETQUOTA user.pdupont (STORAGE 10000)
. OK Completed
```

Est ainsi créée la *quota root* user.pdupont, qui gouverne toute la hiérarchie des boîtes user.pdupont et inférieures (user.pdupont.*). L'utilisateur pdupont sera donc limité à 10000 ko.

La même commande SETQUOTA permet de modifier une *quota root* existante.

Pour connaître la valeur d'une *quota root* existante :

```
. GETQUOTA user.pdupont
* QUOTA user.pdupont (STORAGE 6093 10000)
. OK Completed
```

La réponse indique, outre la valeur maximale autorisée pour le groupe de boîtes (10000 ko), la valeur actuellement utilisée (6093 ko).

La commande `GETQUOTAROOT` peut être exécutée par l'utilisateur lui-même. Son paramètre n'est pas nécessairement une *quota root*, mais n'importe quelle boîte gouvernée par une *quota root*. Elle donne en réponse la *quota root* gouvernante :

```
. GETQUOTAROOT user.pdupont.Corbeille
* QUOTAROOT user.pdupont.Corbeille user.pdupont
* QUOTA user.pdupont (STORAGE 6093 10000)
. OK Completed
```

Bizarrement, il n'y a pas de commande du protocole IMAP pour effacer une *quota root*. Il faut pour cela charcuter les fichiers ([«Overview and Concepts»](#)) :

- Repérer le dossier *configdirectory* :

```
$ cat /etc/imapd.conf
...
configdirectory: /var/imap
...
```

- Dans ce *configdirectory*, on trouve un sous-dossier *quota*, contenant un fichier pour chaque *quota root* existant. Effacer le ou les fichiers correspondants :
- ```
$ cd /var/imap/quota
$ rm user.tavardon
```
- Enfin, pour rendre à la base sa cohérence, il faut reconstruire les quotas. Ça se fait par la commande *quota*, qu'on trouve dans le dossier des binaires du serveur (sur *louis et etienne* : `/usr/cyrus/bin`) :
- ```
$ /usr/cyrus/bin/quota -f
```

Insertion d'un message dans une boîte

```
. APPEND user.pdupont {152}
+ go ahead
From: <Sigmund.Freud@univ-lyon2.fr>
To: <Albert.Einstein@univ-lyon2.fr>
Subject: Les lapins d'Australie
Ne pas les confondre avec les kangourous.

. OK [APPENDUID 896443059 1408] Completed
```

Le nombre donné à la fin de la première ligne est la longueur du message (en tenant compte du fait que les fins de ligne sont `CR + LF`). Cette commande est donc difficile à utiliser pour l'insertion à la main.

Sélection d'une boîte

Pour opérer sur les messages d'une boîte (les examiner, supprimer, ajouter...), il faut d'abord la sélectionner.

```
. SELECT user.pdupont
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
* OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen \*)] * 56
EXISTS
* 21 RECENT
* OK [UNSEEN 1] * OK [UIDVALIDITY 901006991] . OK [READ-WRITE] Completed
```

La réponse se termine normalement par la ligne «OK». Diverses informations sont fournies sur les lignes «* ...», dont le nombre de messages dans la boîte (* 56 EXISTS).

Une fois la boîte sélectionnée, on peut faire diverses opérations décrites plus loin. On peut ensuite si on veut sélectionner une autre boîte. On peut aussi désélectionner une boîte sans en sélectionner une autre par CLOSE.

Une alternative à SELECT est EXAMINE, qui correspond à une ouverture en lecture seule.

Les commandes de manipulation des messages de la boîte sélectionnée

Je suppose pour les besoins des exemples que la boîte sélectionnée (par SELECT) comporte quatre messages, de numéros de séquence 1, 2, 3 et 4, et d'UID 340, 512, 611, et 612.

Listage des messages

```
. SEARCH ALL
* SEARCH 1 2 3 4
. OK Completed
```

Ici, la réponse donne les numéros de séquence des messages, 1, 2, 3 et 4. Il peut être plus intéressant d'obtenir leurs UID, en préfixant la commande par «UID» :

```
. UID SEARCH ALL
* SEARCH 340 512 611 612
. OK Completed
```

On pourra par la suite se servir de ces UID pour désigner les messages.

La clé de recherche «ALL» («tous») n'est pas la seule possible, loin de là. On peut faire des recherches poussées sur la boîte. Exemple :

```
. UID SEARCH SUBJECT "Australie"
* SEARCH 340 611
. OK Completed
. UID SEARCH SUBJECT "Australie" BODY "kangourou"
* SEARCH 340
. OK Completed
. UID SEARCH SUBJECT "Australie" NOT BODY "kangourou"
* SEARCH 611
. OK Completed
```

Parmi les clés de recherche les plus utiles à l'administration :

2:4
Numéro de séquence entre 2 et 4 inclus.

2:*
Idem ; '*' désigne le numéro le plus élevé disponible.

UID 500:700
UID entre 500 et 700 inclus.

SMALLER 4125
Taille du message < 4125 octets

LARGER 4125
Taille du message > 4125 octets

BEFORE 5-Nov-1998
Date interne du message < 5 novembre 1998

SINCE 5-Nov-1998
Date interne du message >= 5 novembre 1998

DELETED
Marqué pour effacement

UNDELETED
Non marqué pour effacement

La taille en question est celle du message entier, entête comprise. La date n'est pas celle du champ «Date:», mais celle d'arrivée du message. Les mois sont : Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec.

N.B. Contrairement à ce qui se passe pour les autres commandes, dans le cas de `SEARCH` le préfixe «UID» («UID SEARCH») agit sur la nature des réponses, et non sur l'interprétation des paramètres.

Lecture de tout ou partie d'un message

```
. FETCH 3 BODY[HEADER]
* 3 FETCH (FLAGS (\Seen) BODY[HEADER] {395}
Return-Path: <kiwi@ornithorynx.au>
Received: from mail.ornithorynx.au
by mailhost.univ-lyon2.fr (8.8.8/jtpda-5.1) with ESMTTP id GAA20695
for <Jean-Marc.Tavardon@univ-lyon2.fr>; Tue, 10 Nov 1998 06:07:28 +0100
(MET)
Date: Tue, 10 Nov 1998 04:31:06 +0100 (MET)
From: <kiwi@ornithorynx.au>
To: <Jean-Marc.Tavardon@univ-lyon2.fr>
Subject: La faune sautante d'Australie
)
. OK Completed
```

Ou en désignant le message par son UID :

```
. UID FETCH 340 BODY[HEADER]
* 3 FETCH (UID 340 BODY[HEADER] {395}
Return-Path: <kiwi@ornithorynx.au>
(...)
)
. OK Completed
```

Le serveur renvoie l'entête, précédé de sa longueur (395).

La description des différentes possibilités pour demander le renvoi de telle ou telle partie est complexe et dépasse l'entendement du rédacteur de cette page (elle permet en particulier au client d'accéder à telle ou telle partie MIME, et de charger le texte principal sans les documents attachés, par exemple). Voici quelques possibilités utiles à l'administration :

FLAGS

Les flags

INTERNALDATE

La date d'arrivée

RFC822.SIZE

La taille

BODY [HEADER]

L'entête

BODY [TEXT]

Le texte

BODY []

L'entête et le texte

BODY [TEXT] <0.500>

Les 500 premiers octets du texte (500 = le nombre d'octets ; BODY [TEXT] <1000.500> correspondrait aux octets numéro 1000 à 1499).

Recopie d'un message vers une autre boîte

```
. COPY 5 Jardin
* 2 EXISTS
* 1 RECENT
. OK [COPYUID 896443059 1407 1410] Completed
```

Le message numéro 5 (numéro de séquence) est copié vers la boîte `Jardin`.

Avec le préfixe «UID», le paramètre est l'UID du message.

Il est possible de copier plusieurs messages d'un coup (ex. : `COPY 1:* Jardin`).

Modification des flags d'un message

C'est surtout utile et nécessaire pour effacer un message :

```
. STORE 3 +FLAGS (\Deleted)
(...)
. OK Completed
```

ou en préfixant par «UID», en indiquant l'UID du message.

On a les trois possibilités suivantes pour l'action de la commande :

FLAGS *liste_complète_de_flags*

La liste indiquée devient la liste des flags du message.

+FLAGS *liste_de_flags_à_ajouter*

Ajoute ces flags à ceux du message.

-FLAGS *liste_de_flags_à_soustraire*

Soustrait ces flags à ceux du message.

Après avoir marqué le message par `\Deleted`, si on veut l'effacer définitivement à jamais, il faut faire une

Purge de la boîte

```
. EXPUNGE  
(...)  
. OK Completed messages effacés définitivement à jamais
```

Tous les messages marqués du flag `\Deleted` sont effacés pour toujours définitivement à jamais.