perlfaq9

Table des matières

1 NAME/NOM

perlfaq9 - Réseau

2 DESCRIPTION

Cette section traite des questions relatives aux aspects réseau, à internet et un peu au web.

2.1 Quelle est la forme correcte d'une réponse d'un script CGI ?

(Alan Flavell <flavell+www@a5.ph.gla.ac.uk> répond...)

La Common Gateway Interface (CGI) spécifie une interface logicielle entre un programme ("le script CGI") et un serveur web (HTTPD). Cette interface n'est pas spécifique à Perl et possède ses propres FAQ et tutoriels ainsi qu'un forum de discussions comp.infosystems.www.authoring.cgi.

Les spécifications CGI sont présentées dans la RFC d'information 3875 (http://www.ietf.org/rfc/rfc3875).

D'autres documentations intéressantes sont listées sur http://www.perl.org/CGI_MetaFAQ.html.

Ces FAQ Perl répondent à quelques problèmes CGI. Mais nous incitons fortement les programmeurs Perl à utiliser le module CGI.pm qui fait attention à tous les détails pour eux.

La ressemblance entre les en-têtes d'une réponse CGI (définis dans les spécifications CGI) et ceux d'une réponse HTTP (tels que défini dans les spécifications HTTP, RFC2616) sontest volontaire mais peut parfois amener à certaines confusions.

Les spécifications CGI définissent deux sortes de scripts : les cripts "Parsed Header" (dont les en-têtes sont traités) et les scripts "Non Parsed Header" ou NPH (dont les en-têtes ne sont pas traités). Chercher dans la documentation de votre serveur pous savoir ce qu'il accepte. Les scripts "Parsed Header" sont plus simples pour plusieurs raisons. Les spécifications CGI autorisent n'importe quelle représentation usuelle des fins de ligne dans la réponse CGI (c'est au serveur HTTP de produire à partir de cette réponse, une réponse HTTP correcte). Donc un "\n" écrit en mode texte est techniquement correct et recommandé. Les scripts NPH sont plus exigeants : ils doivent produire une réponse contenant un ensemble d'en-têtes HTTP complets et corrects. Or les spécifications HTTP exigent des lignes terminées par un retour chariot (carraige-return) puis un saut de ligne (line-feed), c'est à dire les valeurs ASCII \015 et \012 écrites en mode binaire.

L'utilisation de CGI.pm permet d'être indépendant de la plateforme, même si celle-ci est un système EBCDIC. CGI.pm choisit pour les fins de ligne la représentation appropriée ($CGI::CRLF) et positionne le mode binaire comme il faut.

2.2 Mon script CGI fonctionne en ligne de commandes mais pas depuis un navigateur. (500 Server Error)

Plusieurs problèmes peuvent en être la cause. Consultez le guide "Troubleshooting Perl CGI scripts" ("Problème de scripts CGI en Perl") sur :

```
http://www.perl.org/troubleshooting_CGI.html
```

Ensuite, si vous pouvez montrer que vous avez lu les FAQ et que votre problème n'est pas quelque chose de simple dont on trouve facilement la réponse, vous recevrez sans doute des réponses courtoises et utiles en postant votre question sur comp.infosystems.www.authoring.cgi (si elle a un quelconque rapport avec les protocoles HTTP et CGI). Les questions qui semblent porter sur Perl mais qui en fait sont liées à CGI et qui sont postées sur comp.lang.perl.misc sont souvent mal accueillies.

Les FAQ utiles et les guides d'aide sont listés dans la Meta FAQ CGI :

```
http://www.perl.org/CGI_MetaFAQ.html
```

2.3 Comment faire pour obtenir de meilleurs messages d'erreur d'un programme CGI ?

Utilisez le module CGI::Carp. Il remplace `warn` et `die`, plus les fonctions normales `carp`, `croak`, et `confess` du module Carp qui sont des versions plus parlantes et sûres. Les erreurs continuent à être envoyées vers le fichier normal des erreurs du serveur.

```
use CGI::Carp;
warn "This is a complaint";
die "But this one is serious";
```

L'utilisation suivante de CGI::Carp redirige également les erreurs vers un fichier de votre choix, mais aussi les avertissements durant la phase de compilation en étant place dans un bloc BEGIN.

```
BEGIN {
    use CGI::Carp qw(carpout);
    open(LOG, ">>/var/local/cgi-logs/mycgi-log")
        or die "Unable to append to mycgi-log: $!\n";
    carpout(*LOG);
}
```

Vous pouvez vous arrangez pour que les erreurs fatales soient retournées au navigateur client, ceci vous permettant d'obtenir un meilleur debogage, mais pouvant paraître confus pour l'utilisateur final.

```
use CGI::Carp qw(fatalsToBrowser);
die "Bad error here";
```

Si l'erreur se produit avant même que vous ayez produit les en-têtes HTTP en sortie, le module essayera de les générer pour éviter une erreur 500 du serveur. Les avertissements normaux continueront à être envoyés vers le fichier de log des erreurs du serveur (ou là où vous les avez envoyées via `carpout`) préfixés par le nom du script et la date.

2.4 Comment enlever les balises HTML d'une chaîne ?

La meilleure façon (mais pas obligatoirement la plus rapide) est d'utiliser HTML::Parser disponible sur le CPAN. Un autre moyen généralement correct est l'utilisation de HTML::FormatText qui non seulement retire le HTML mais aussi essaye de réaliser un petit formatage simple du texte brut résultant.

Plusieurs personnes essayent une approche simpliste utilisant des expressions rationnelles, comme `s/<.*?>//g`, mais ceci ne fonctionne pas correctement dans de nombreux cas car les marqueurs HTML peuvent continuer après des sauts de lignes, ils peuvent contenir des < ou des > entre guillemets, ou des commentaires HTML peuvent être présents. De plus, ces personnes oublient de convertir les entités comme `<` par exemple.

Voici une "solution-basique", qui fonctionne avec la plupart des fichiers :

```
#!/usr/bin/perl -p0777
s/<(?:[^>'"]*|(['"]).*?\1)*>//gs
```

Si vous souhaitez une solution plus complète, regardez le programme striphtml se décomposant en 3 étapes sur http://www.cpan.org/authors/Tom_Christiansen/scripts/striphtml.gz.

Voici quelques pièges auxquels vous devriez penser avant de choisir une solution :

```
<IMG SRC = "foo.gif" ALT = "A > B">

<IMG SRC = "foo.gif"
     ALT = "A > B">

<!-- <Un commentaire> -->

<script>if (a<b && a>c)</script>

<# Just data #>

<![INCLUDE CDATA [ >>>>>>>>>>>> ]]>
```

Si les commentaires HTML incluent d'autres balises, ces solutions échoueraient aussi sur un texte tel que celui-ci :

```
<!-- Cette section est en commentaire.
    <B>You can't see me!</B>
-->
```

2.5 Comment extraire des URL ?

Vous pouvez facilement extraire toutes sortes d'URL d'un document HTML en utililisant `HTML::SimpleLinkExtor` qui gèrent les ancres, les images, les objets, les cadres et plein d'autres balises qui peuvent contenir des URL. Si vous avez besoin de quelque chose de plus complexe, vous pouvez créer votre propre sous-classe en héritant de `HTML::LinkExtor` ou de `HTML::Parser`. Vous pouvez même utilisez `HTML::SimpleLinkExtor` comme base de travail pour répondre à vos besoins spécifiques.

Vous pouvez utiliser `URI::Find` pour extraire des URL d'un document texte quelconque.

Des solutions moins complètes basées sur des expressions rationnelles peuvent économiser du temps de calcul si vous êtes sûr que l'entrée est simple. Voici une solution, proposée par Tom Christiansen, qui va 100 fois plus vite que la plupart des solutions fournies par les modules mais qui n'extraient des URL que des ancres (les balises A) dont le seul attribut est un HREF :

```
#!/usr/bin/perl -n00
# qxurl - tchrist@perl.com
print "$2\n" while m{
    < \s*
      A \s+ HREF \s* = \s* (["']) (.*?) \1
    \s* >
}gsix;
```

2.6 Comment télécharger un fichier depuis la machine d'un utilisateur ? Comment ouvrir un fichier d'une autre machine ?

Dans ce cas, télécharger signifie utiliser la fonctionalité d'envoi de fichier via un formulaire HTML. Vous autorisez ainsi l'utilisateur à choisir un fichier qui sera envoyé à votre serveur web. Pour vous, c'est un téléchargement alors que, pour l'utilisateur, c'est un envoi. Peu importe son nom, vous pourrez le faire en utilisant ce qu'on appele l'encodage **multipart/form-data**. Le module CGI.pm (qui est un module standard de la distribution Perl) propose cette fonctionalité dans via la méthode start_multipart_form() qui n'est pas la même que la méthode startform().

Pour des exemples de codes et des informations supplémentaires, voir la section parlant du téléchargement de fichiers dans la documentation du module CGI.pm.

2.7 Comment faire un menu pop-up en HTML ?

Utiliser les tags **SELECT** et **OPTION**. Le module CGI.pm (disponible au CPAN) support cette fonctionnalité, ainsi que de nombreuses autres, incluant quelques-unes qui se synthétisent intelligemment d'elles-mêmes.

2.8 Comment récupérer un fichier HTML ?

Utiliser le module LWP::Simple disponible au CPAN, qui fait partie de l'excellent package libwww-perl (LWP). D'un autre côté, si vous avez le navigateur en mode texte lynx installé sur votre système, il n'est pas mauvais de faire :

```
$html_code = `lynx -source $url`;
$text_data = `lynx -dump $url`;
```

Les modules libwww-perl (LWP) du CPAN fournissent une façon plus puissante de le faire. Ils n'ont pas besoin de lynx, mais tout comme lynx, ils peuvent fonctionner à travers les serveurs mandataires (proxy, NDT) :

```
# version la plus simple
use LWP::Simple;
$content = get($URL);

# ou affiche du HTML depuis un URL
use LWP::Simple;
getprint "http://www.linpro.no/lwp/";
```

```
# ou affiche de l'ASCII depuis le HTML d'un URL
# nécessite aussi le paquetage HTML-Tree du CPAN
use LWP::Simple;
use HTML::Parser;
use HTML::FormatText;
my ($html, $ascii);
$html = get("http://www.perl.com/");
defined $html
    or die "Can't fetch HTML from http://www.perl.com/";
$ascii = HTML::FormatText->new->format(parse_html($html));
print $ascii;
```

2.9 Comment automatiser la soumission d'un formulaire HTML ?

Si vous souhaitez faire quelque chose complexe comme parcourir plusieurs pages et formulaires d'un site web, vous pouvez utiliser le module `WWW::Mechanize`. Voir sa documentation pour plus de détails.

Si vous soumettez des valeurs en utilisant la méthode GET, créez un URL et codez le formulaire en utilisant la méthode `query_form` :

```
use LWP::Simple;
use URI::URL;

my $url = url('http://www.perl.com/cgi-bin/cpan_mod');
$url->query_form(module => 'DB_File', readme => 1);
$content = get($url);
```

Si vous utilisez la méthode POST, créez votre propre agent utilisateur et codez le contenu de façon appropriée.

```
use HTTP::Request::Common qw(POST);
use LWP::UserAgent;

$ua = LWP::UserAgent->new();
my $req = POST 'http://www.perl.com/cgi-bin/cpan_mod',
               [ module => 'DB_File', readme => 1 ];
$content = $ua->request($req)->as_string;
```

2.10 Comment décoder ou créer ces %-encodings sur le web ?

Si vous écrivez un script CGI, vous devriez utiliser le module CGI,pm qui vient avec perl ou un autre module équivalent. Le module CGI décode automatiquement les requêtes et propose la fonction encode() pour gérer l'encodage.

La meilleure source d'informations au sujet de l'encodage des URI est la RFC 2396. En principe, les substitutions suivantes le font bien :

```
s/([^\w()'*~!.-])/sprintf '%%%02x', ord $1/eg;   # encodage

s/%([A-Fa-f\d]{2})/chr hex $1/eg;                # décodage
s/%([[:xdigit:]]{2})/chr hex $1/eg;              # idem
```

En revanche, vous ne devriez les appliquer qu'à des composants individuels d'un URI et non à l'URI en entier. Sinon vous risquez de perdre de l'information et donc de rater quelque chose. Si vous ne comprenez pas pourquoi, pas de panique. Consultez la section 2 de la RFC qui donne probablement les meilleurs explications à ce sujet.

La RFC 2396 contient aussi plein d'autres informations intéressantes et en particulier des expressions rationnelles permettant de découper en composant un URI quelconque (annexe B).

2.11 Comment rediriger le navigateur vers une autre page ?

Spécifiez l'URL complet de la page de destination (même si elle est sur le même serveur). C'est l'une des deux sortes de réponses "Location:" prévues par les spécifications CGI pour un script "Parser Header". L'autre sorte qui retourne "Location:" avec un URL sous la forme d'un chemin absolu (sans le schéma ni le nom du serveur) est résolu en interne par le serveur sans utiliser la redirection HTTP. En tous cas, les spécifications CGI n'autorisent pas les URL relatifs.

L'utilisation de CGI.pm est fortement recommandée. L'exemple ci-dessous montre une redirection avec un URL complet. Cette redirection est gérée par le navigateur web.

```
use CGI qw/:standard/;

my $url = 'http://www.cpan.org/';
print redirect($url);
```

Ce deuxième exemple montre une redirect avec un URL sous la forme d'un chemin absolu. Cette redirect est gérée par le serveur web local.

```
my $url = '/CPAN/index.html';
print redirect($url);
```

Si vous souhaitez le coder vous-même directement, c'est faisable comme suit (le "\n" final est montré séparément pour être plus clair). que l'URL soit complet ou sous la forme d'un simple chemin absolu.

```
print "Location: $url\n";   # en-tête de la réponse CGI
print "\n";                 # fin de l'en-tête
```

2.12 Comment mettre un mot de passe sur mes pages Web ?

Pour que l'authentification soit active sur votre serveur web, il faut le configurer pour cela. Cette configuration diffèrent selon le serveur web – apache ne fait pas comme iPlanet qui ne fait pas comme IIS. Consultez la documentation de votre serveur web pour connaître les détails de configuration de ce serveur.

2.13 Comment éditer mes fichiers .htpasswd et .htgroup en Perl ?

Les modules HTTPD::UserAdmin et HTTPD::GroupAdmin proposent une interface orientée objet cohérente pour ces fichiers, quelle que soit la façon dont ils sont stockés. Les bases de données peuvent être du texte, une dbm, une Berkley DB ou n'importe quelle autre base de données avec un pilote compatible DBI. HTTPD::UserAdmin accepte les fichiers utilisés par les mécanismes d'authentification 'Basic' et 'Digest'. Voici un exemple :

```
use HTTPD::UserAdmin ();
HTTPD::UserAdmin
      ->new(DB => "/foo/.htpasswd")
      ->add($username => $password);
```

2.14 Comment être sûr que les utilisateurs ne peuvent pas entrer de valeurs dans un formulaire qui font faire de vilaines choses à mon script CGI ?

Voyez les références à la sécurité dans la Meta FAQ CGI :

```
http://www.perl.org/CGI_MetaFAQ.html
```

2.15 Comment analyser un en-tête de mail ?

Pour une solution rapide et peu propre, essayez cette solution dérivée de split in *perlfunc* :

```
$/ = '';
$header = <MSG>;
$header =~ s/\n\s+/ /g;      # fusionne les lignes fractionnées
%head = ( UNIX_FROM_LINE, split /^([-\w]+):\s*/m, $header );
```

Cette solution ne fonctionne pas correctement si, par exemple, vous essayez de mettre à jour toutes les lignes reçues. Une approche plus complète consiste à utiliser le module Mail::Header du CPAN (faisant parti du paquetage MailTools).

2.16 Comment décoder un formulaire CGI ?

(contribution de brian d foy)

Utilisez la module CGI.pm qui vient avec Perl. Il est rapide, simple et effectue le travail nécessaire pour s'assurer que tout se passe correctement. Il gère les requêtes GET, POST et HEAD, les formulaires multipart, les champs multi-valués, les combinaisons chaîne de requête et données de flux, et plein d'autres choses dont vous ne voulez probablement pas vous préoccuper.

Et tout cela d'une manière on ne peut plus `simple` : le module CGI analyse automatiquement les entrées et rend accessible chacune des valeurs via la fonction `param()`.

```
use CGI qw(:standard);

my $total = param( 'prix' ) + param( 'transport' );
my @items = param( 'item' ); # valeurs multiples, pour un même champ
```

Si vous préférez une interface orientée objet, CGI.pm sait aussi le faire.

```
use CGI;

my $cgi = CGI->new();

my $total = $cgi->param( 'prix' ) + $cgi->param( 'transport' );

my @items = $cgi->param( 'item' );
```

Vous pouvez aussi essayer le module CGI::Minimal qui est version allégée pour faire la même chose. D'autres modules CGI::* de CPAN peuvent aussi mieux vous convenir.

Beaucoup de gens essaye d'écrire leur propre décodeur (ou recopie celui d'un autre programme) et tombe ensuite dans l'un des nombreux pièges de cette tâche. Il est plus simple et moins embêtant d'utiliser CGI.pm.

2.17 Comment vérifier la validité d'une adresse électronique ?

Vous ne pouvez pas, du moins pas en temps réel. Dommage, hein ?

Sans envoyer un mail à cette adresse pour constater qu'il y a un être humain à l'autre bout pour vous répondre, vous ne pouvez pas déterminer si une adresse est valide. Même si vous appliquez l'en-tête standard d'email, vous pouvez rencontrer des problèmes, car il existe des adresses valides qui ne sont pas compatibles avec la RFC-822 (le standard des en-têtes des mails), et inversement il existe des adresses qui ne sont pas délivrables et qui sont compatibles.

Vous pouvez utiliser les modules Email::Valid ou RFC::RFC822::Address qui vérifient le format de l'adresse bien que cela ne garantit en rien que l'adresse est valide (c'est à dire qu'un message envoyé ne sera pas refusé). Les modules comme Mail::CheckUser et Mail::EXPN interogent le système de noms de domaines et les serveurs de messagerie pour tenter d'en apprendre plus mais cela ne marche pas partout – surtout si l'administrateur du domaine gère consciencieusement la sécurité de son site.

Beaucoup sont tentés d'éliminer les adresses email invalides en se basant sur une expression régulière simple, comme `/^[\w.-]+\@(?:[\w-]+\.)+\w+$/`. C'est une très mauvaise idée. Car on rejette ainsi beaucoup d'adresses valides et que cela ne garantit en rien que les adresses acceptées sont effectivement délivrables, ce n'est donc pas conseillé. À la place, regardez plutôt http://www.cpan.org/authors/Tom_Christiansen/scripts/ckaddr.gz qui effectivement vérifie une compatibilité complète avec les spécifications RFC (excepté les commentaires imbriqués), vérifie qu'il ne s'agit pas d'une adresse que vous ne désirez pas (cad, Bill Clinton ou votre responsable de compte mail), puis s'assure que le nom d'hôte donné peut être trouvé dans les enregistrements MX du DNS. Ce n'est pas très rapide, mais ça marche pour ce que ça essaye de faire.

Notre meilleur conseil pour vérifier l'adresse de quelqu'un est de lui faire entrer deux fois son adresse, tout comme vous le faites pour changer un mot de passe. Ceci élimine habituellement les fautes de frappe. Si les deux versions sont égales, envoyez un courrier à cette adresse avec un message personnel ayant cette allure :

```
Dear someuser@host.com,
```

```
Please confirm the mail address you gave us Wed May  6 09:38:41
MDT 1998 by replying to this message.  Include the string
"Rumpelstiltskin" in that reply, but spelled in reverse; that is,
start with "Nik...".  Once this is done, your confirmed address will
be entered into our records.
```

Si vous recevez le message et s'ils ont suivi vos indications, vous pouvez être raisonnablement assuré que l'adresse est réelle.

Une stratégie proche moins ouverte à la tromperie est de leur donner un PIN (numéro d'identification personnel). Enregistrez l'adresse et le PIN (le mieux est qu'il soit aléatoire) pour un traitement ultérieur. Dans le message que vous envoyez, demandez-leur d'inclure le PIN dans leur réponse. Mais si le message rebondit, ou est inclus automatiquement par un script "vacation" (en vacances), il sera là de toute façon. Il est donc plus efficace de leur demander de renvoyer un PIN légèrement modifié, par exemple inversé, ou une unité ajoutée à chaque chiffre, etc.

2.18 Comment décoder une chaîne MIME/BASE64 ?

Le module MIME::Base64 (disponible sur CPAN) gère cela ainsi que l'encodage MIME/QP. Décoder du BASE64 devient alors aussi simple que :

```
use MIME::Base64;
$decoded = decode_base64($encoded);
```

Le paquetage MIME-tools (disponible sur CPAN) propose ces extractions avec en sus le décodage des documents attachés encodés en BASE64, tout cela directement depuis un message e-mail.

Si la chaîne à décoder est courte (moins de 84 caractères), une approche plus directe consiste à utiliser la fonction unpack() avec le formatage "u" après quelques translations mineures :

```
tr#A-Za-z0-9+/##cd;                 # supprime les caractères non base-64
tr#A-Za-z0-9+/# -_#;                # convertit dans le format uuencode
$len = pack("c", 32 + 0.75*length); # calcule la longueur en octets
print unpack("u", $len . $_);       # uudécode et affiche
```

2.19 Comment renvoyer l'adresse électronique de l'utilisateur ?

Sur les systèmes supportant getpwuid, et donc la variable $<, ainsi que le module Sys::Hostname (qui fait partie de la distribution standard de Perl), vous pouvez probablement essayer d'utiliser quelque chose comme ceci :

```
use Sys::Hostname;
$address = sprintf('%s@%s', getpwuid($<), hostname);
```

La politique de la compagnie sur les adresses email peut signifier que ceci génère des adresses que le système de mail de la compagnie n'acceptera pas, ainsi vous devriez demander les adresses email des utilisateurs quand ceci compte. Qui plus est, tous les systèmes sur lesquels fonctionne Perl n'acceptent pas ces informations comme sur Unix.

Le module Mail::Util du CPAN (faisant partie du package MailTools) procure une fonction mailaddress() qui essaye de créer l'adresse email d'un utilisateur. Il effectue une démarche plus intelligente que le code précédent, utilisant des informations fournies quand le module a été installé, mais cela peut rester incorrect. Encore une fois, la meilleure manière est souvent de simplement poser la question à l'utilisateur.

2.20 Comment envoyer un mail ?

Utilisez directement le programme `sendmail` :

```
open(SENDMAIL, "|/usr/lib/sendmail -oi -t -odq")
                    or die "Can't fork for sendmail: $!\n";
print SENDMAIL <<"EOF";
From: User Originating Mail <me\@host>
To: Final Destination <you\@otherhost>
Subject: A relevant subject line
```

```
    Body of the message goes here after the blank line
    in as many lines as you like.
    EOF
    close(SENDMAIL)     or warn "sendmail didn't close nicely";
```

L'option **-oi** empêche sendmail d'interpréter une ligne constituée d'un seul point comme une "fin de message". L'option **-t** lui dit d'utiliser les en-têtes pour décider à qui envoyer le message, et **-odq** lui dit de placer le message dans la file d'attente. Cette dernière option signifie que votre message ne sera pas immédiatement envoyé, donc ne la mettez pas si vous voulez un envoi immédiat.

Alternativement, des approches moins pratiques comprennent l'appel direct à mail (parfois appelé mailx) ou la simple ouverture du port 25 pour avoir une conversation intime rien qu'entre vous et le démon SMTP distant, probablement sendmail.

Ou vous pourriez utiliser le module Mail::Mailer du CPAN :

```
    use Mail::Mailer;

    $mailer = Mail::Mailer->new();
    $mailer->open({ From    => $from_address,
                    To      => $to_address,
                    Subject => $subject,
                  })
        or die "Can't open: $!\n";
    print $mailer $body;
    $mailer->close();
```

Le module Mail::Internet utilise Net::SMTP qui est moins Unix-centrique que Mail::Mailer, mais moins fiable. Évitez les commandes SMTP crues. Il y a de nombreuses raisons pour utiliser un agent de transport de mail comme sendmail. Celles-ci comprennent la mise en file d'attente, les enregistrement MX et la sécurité.

2.21 Comment utiliser MIME pour attacher des documents à un mail ?

Cette réponse est directement extraite de la documentation du module MIME::Lite. Créez un message multipart (c'est à dire avec des documents attachées).

```
use MIME::Lite;

### Création d'un nouveau message multipart
$msg = MIME::Lite->new(
             From    =>'me@myhost.com',
             To      =>'you@yourhost.com',
             Cc      =>'some@other.com, some@more.com',
             Subject =>'A message with 2 parts...',
             Type    =>'multipart/mixed'
             );

### Ajout de parties (chaque "attach" a les mêmes arguments que "new"):
$msg->attach(Type     =>'TEXT',
             Data     =>"Here's the GIF file you wanted"
             );
$msg->attach(Type     =>'image/gif',
             Path     =>'aaa000123.gif',
             Filename =>'logo.gif'
             );

$text = $msg->as_string;
```

MIME::Lite propose aussi une méthode pour envoyer le messages.

```
$msg->send;
```

Par défaut, il utilise `sendmail` mais on peut le paramètrer pour qu'il utilise SMTP via *Net::SMTP*.

2.22 Comment lire du courrier ?

Vous pourriez utiliser le module Mail::Folder de CPAN (faisant partie du paquetage MailFolder) ou le module Mail::Internet de CPAN (faisant partie du paquetage MailTools), toutefois, un module est souvent un marteau pour écraser une mouche. Voici un trieur de mail.

```
#!/usr/bin/perl

my(@msgs, @sub);
my $msgno = -1;
$/ = '';                    # lit un paragraphe
while (<>) {
  if (/^From/m) {
    /^Subject:\s*(?:Re:\s*)*(.*)/mi;
    $sub[++$msgno] = lc($1) || '';
  }
  $msgs[$msgno] .= $_;
}
for my $i (sort { $sub[$a] cmp $sub[$b] || $a <=> $b } (0 .. $#msgs)) {
  print $msgs[$i];
}
```

Ou de façon plus succincte,

```
#!/usr/bin/perl -n00
# bysub2 - awkish sort-by-subject
BEGIN { $msgno = -1 }
$sub[++$msgno] = (/^Subject:\s*(?:Re:\s*)*(.*)/mi)[0] if /^From/m;
$msg[$msgno] .= $_;
END { print @msg[ sort { $sub[$a] cmp $sub[$b] || $a <=> $b } (0 .. $#msg) ] }
```

2.23 Comment trouver mon nom de machine / nom de domaine / mon adresse IP ?

(contribution de brian d foy)

Le module Net::Domain, qui fait partie de la distribution standard depuis Perl 5.7.3, peut vous fournir le nom d'hôte complet (fully qualified domain name ou FQDN), le nom d'hôte (ou de machine) ou le nom de domaine.

```
use Net::Domain qw(hostname hostfqdn hostdomain);

my $host = hostfqdn();
```

Le module `Sys::Hostname`, inclus dans la distribution standard depuis Perl 5.6, peut aussi vous donner le nom d'hôte.

```
use Sys::Hostname;

$host = hostname();
```

Pour obtenir l'adresse IP, vous pouvez faire appel à la fonction prédéfinie <gethostbyname> pour transformer le nom d'hôte en un nombre. Pour convertir ce nombre en une adresse classique (a.b.c.d) que la plupart des gens attendent, utilisez la fonction `inet_ntoa` du module `Socket` qui fait partie de perl.

```
use Socket;

my $address = inet_ntoa(
  scalar gethostbyname( $host || 'localhost' )
  );
```

2.24 Comment récupérer un article de news ou les groupes actifs ?

Utilisez les modules Net::NNTP ou News::NNTPClient, tous les deux disponibles sur le CPAN. Ceux-ci peuvent rendre des tâches comme la récupération de la liste des groupes de news aussi simple que :

```
perl -MNews::NNTPClient
  -e 'print News::NNTPClient->new->list("newsgroups")'
```

2.25 Comment récupérer/envoyer un fichier par FTP ?

LWP::Simple (disponible sur le CPAN) peut récupérer mais pas envoyer. Net::FTP (aussi disponible sur le CPAN) est plus complexe, mais peut envoyer aussi bien que récupérer.

2.26 Comment faire du RPC en Perl ?

(contribution de brian d foy)

Utilisez l'un des modules RPC disponibles sur CPAN (http://search.cpan.org/search?query=RPC&mode=all).

3 AUTEUR ET COPYRIGHT

4 TRADUCTION

4.1 Version

Cette traduction française correspond à la version anglaise distribuée avec perl 5.8.8. Pour en savoir plus concernant ces traductions, consultez http://perl.enstimac.fr/.

4.2 Traducteur

Traduction initiale : Aymeric Barantal <Aymeric.Barantal@grolier.fr>. Mise à jour : Paul Gaborit <paul.gaborit at enstimac.fr>.

4.3 Relecture

Régis Julié <Regis.Julie@cetelem.fr>, Roland Trique <roland.trique@uhb.fr> (mise à jour), Gérard Delafond.

5 À propos de ce document

Ce document est la traduction française du document orginal distribué avec perl. Vous pouvez retrouver l'ensemble de la documentation française Perl (éventuellement mise à jour) en consultant l'URL <http://perl.enstimac.fr/>.

Ce document PDF a été produit Paul Gaborit. Si vous utilisez la version PDF de cette documentation (ou une version papier issue de la version PDF) pour tout autre usage qu'un usage personnel, je vous serai reconnaissant de m'en informer par un petit message <mailto:Paul.Gaborit@enstimac.fr>.

Si vous avez des remarques concernant ce document, en premier lieu, contactez la traducteur (vous devriez trouver son adresse électronique dans la rubrique TRADUCTION) et expliquez-lui gentiment vos remarques ou critiques. Il devrait normalement vous répondre et prendre en compte votre avis. En l'absence de réponse, vous pouvez éventuellement me contacter.

Vous pouvez aussi participer à l'effort de traduction de la documentation Perl. Toutes les bonnes volontés sont les bienvenues. Vous devriez trouver tous les renseignements nécessaires en consultant l'URL ci-dessus.