迷惑メールの報告スクリプト

2013年10月29日
色々試してencode()で MIME-HEADERを作る時の条件判明。気付かんかったorz

2013年10月28日
文言の微修正。

2013年10月25日
いまさらなんですが修正したはずのところが修正されていないことに気が付いて、あわてて再編集。どうせなので今の環境に合わせて加筆修正する。

これは何？

これは、寝込んでいたあいだに送られてきた迷惑メールの多さに頭に来た筆者が、財団法人日本データ通信協会迷惑メール相談センターへの自動報告のために記述したPerlスクリプトのメモ。

始まりは溜まりに溜まった迷惑メール

寝込んでたんですよ。で、やっと回復してメール見たら…送ってきた奴ら、必ず痛い目見させてやる。

その他

平成20年12月1日
特定商取引法改正に伴い、財団法人 日本産業協会 迷惑メール情報提供受付アドレスが、従来の mailagain@nissankyo.jp から spam-in@nissankyo.jp に変更となりました。

やったこと

迷惑メールの報告先確認

公の機関で迷惑メールの報告を行えるところが２箇所あります。

- 総務省管轄　　→　財団法人 日本データ通信協会 迷惑メール相談センター
- ~~経済産業省管轄　→　財団法人 日本産業協会~~ 迷惑メール情報提供受付

日本データ通信協会の方は、報告に際して問題のメールを添付の形式で一括送信OKのようです。 ~~日本産業協会の方は、転送しろ、としか書かれていないので基本、一件ずつの報告になるようです。~~ 日本産業協会の方はまっとうに機能していないようです。エラーメールが返ってばかり。

MIME-toolsの導入

一つずつ手で処理してられないので、添付メールを生成するスクリプトを書くことにしました。 うちの場合、メールはすべて FreeBSDのサーバ機でさばいています。ですのでPerlのスクリプトを書いて処理

させるのが楽であろうと判断。で、ちょっとWebを見るとMIME-tools を使った例をたくさん見ることができたのでこれを使うことにしました portsで p5-MIME-tools をさっくり導入 Perlのバージョンは 5.14.4。

所要時間は、調べものをして約２時間です。…あ、手作業で送ったほうが早かったんじゃない？とおっしゃる方、この作業を継続するならスクリプトにしたほうが楽ですよ。

参照サイト

Perl5.8.8以降であれば多国語対応しているので、昔のようにJcode.plやら何やらをそろえる必要が無いねー、とか考えていたら。やっぱり色々ありました。以下のリンクを見て何とか解決。感謝。

- Perl 5.8.x における日本語コード変換のメモ
- Perlメモ/日本語の扱い
- Encode
- Perl 5.8.x Unicode関連
- livedoor Developers Blog モブログに潜んでいる不具合

何に嵌ったかというとEUC→JISへの変換です。ちょっと癖があるというか。ちゃんと理解していないとまったく訳が分からない現象に悩まされることになります。

日本データ通信協会用メール送信スクリプト

日本データ通信協会へは複数メールを添付した形式でメールを送ります。

```
use Encode;
use MIME::Entity;
use MIME::Words qw (:all);

my $localencode = "euc-jp";
my $maildir     = "$ENV{HOME}/Maildir/.WORK.TEIKYO/cur";
my $from        = 'hogehoge@hogehoge.jp';
my $to          = 'meiwaku@dekyo.or.jp';
my $Subject     = "表示義務に違反するメールがありましたので、情報提供いたします";
my $messageText = "財団法人 日本データ通信協会"."\n"
                 ."迷惑メール相談センター" . "\n"
                 ."担当者 様" . "\n"
                 ."\n"
                 ."\n"
                 ."表示義務に違反するメールを受信しましたので情報提供いたします。" .
"\n"
                ."本メールに添付したものが提供させていただきます表示義務違反メールです。
" . "\n"
                 ."\n"
                 ."大変お手数ではありますが対応いただきたく、よろしくお願いいたしま
す。" . "\n";
my $mailmsg;

$Subject     = encode( "jis", decode($localencode, $Subject)     );
$messageText = encode( "jis", decode($localencode, $messageText) );
```

```
## メール枠生成
$mailmsg = MIME::Entity->build(
                               From    => $from
                              ,Sender  => $from
                              ,To      => $to
                              ,Subject => encode_mimeword( $Subject,
'B','iso-2022-jp')
                              ,Type    => 'multipart/mixed'
                            );
## 本文
$mailmsg->attach(
                Type     => 'text/plain; charset=iso-2022-jp'
               ,Encoding => '7bit'
               ,Data     => $messageText
           );

## 添付処理
  opendir(DH, $maildir);
  @maillist = readdir(DH);
  closedir(DH);
  $cnt=1;
  while(<@maillist>) {
    if (($_ ne ".")&&($_ ne "..")) {
      $mailmsg->attach(
                      Type        => 'message/rfc822'
                     ,Encoding    => '7bit'
                     ,Description => "attachment"
                     ,Path        => "$maildir/$_"
                     ,Filename    => sprintf("mwkmail%03i.eml",$cnt)
                  );
      $cnt++;
    }
  }

## メール送信
$mailmsg->send('sendmail');
```

$maildir で指定するディレクトリ（ここでは$HOME/Maildir/.WORK.TEIKYO/cur□中のメールファイルを全て添付とし□ $to で示すあて先（迷惑メール相談センターの受付メールアドレス）に送信します□$from は、送信者(筆者)のメールアドレスになります。 テストする際には$fromも$toも自分のメールアドレスにするように。受信結果が正しいことを必ず確認しなければいけません。 また、ひと頃昔なら$fromに詐称したアドレスを書けたかもしれませんが、最近ではそれやるとメール送信できないプロバイダが多いかもしれません。

筆者はMaildir形式を使っているので、１メール１ファイルになります□readdir()でメールの一覧を取り出しています□mbox形式の場合は1ファイルに複数のメールが記録されますので適宜メールを切り出す処理を書く必要があります。

ソース中の$localencodeには、このPerlスクリプトで使っているエンコーディングを指定してください。
※うちではeuc-jpを使ってます。
decode()でPerlスクリプトのeuc-jpで表現された文字列をPerl内部表現へ変換しています。そし

てencode()で内部表現の文字列をjis表現の文字列へ変換します。

ちなみに、このスクリプトをそのまま使ったわけではないです。 例えば添付が１００個とか付いたメールはとても重くて開くことが困難だと思います。適宜添付数を調整したり、同じ送信元同じ内容のメール(重複したメール)は添付対象からはずす、送付済みのメールは送信対象からはずす、などの処理がいりますね。

ちょっと嵌った

MIME-Toolsのencode_mimeword()を使いMIMEを使ったSubjectヘッダに変換しています。
今のPerlならこちらを使わずともencode("MIME-HEADER-ISO_2022_JP", JIS表現の文字列) でいけるかもしれません。ただ、何度かテストしたときにうまく変換できないことがあったのでうちではencode_mimeword()を使っています。

…だったんですけどね、どうも私の使い方がダメダメだったようです。
encode()でエンコードする形式が " MIME-HEADER-xxx"の場合でも変換元はPerl内部形式を喰わさなきゃいけません… 具体的には encode("MIME-HEADER-ISO_2022_JP", Perl内部表現の文字列) にします。以下の例の様に。

a.pl

```
use Encode;
use MIME::Words qw (:all);

my $subject   = "表示義務に違反するメールがありましたので、情報提供いたします";
my $perl      = decode("euc-jp", $subject);                    ## euc-jp
をPerl内部形式に
my $jis       = encode("jis",    $perl);                       ## Perl内部
形式をJISに
my $utf8      = encode("utf-8",  $perl);                       ## Perl内部
形式をutf-8に

my $mime1jis  = encode_mimeword( $jis,  'B', 'iso-2022-jp'); ## $perl
を使っちゃダメ
my $mime1utf8 = encode_mimeword( $utf8, 'B', 'utf-8');       ## $perl
を使っちゃダメ
my $mime2jis  = encode('MIME-HEADER-ISO_2022_JP', $perl);    ## $jis
を使っちゃダメ
my $mime2utf8 = encode('MIME-HEADER',             $perl);    ## $utf8
を使っちゃダメ

print "   org      [" . $subject   ."]\n";
print "  MIME JIS  [" . $mime1jis  ."]\n";
print "Encode JIS  [" . $mime2jis  ."]\n";
print "  MIME UTF-8[" . $mime1utf8 ."]\n";
print "Encode UTF-8[" . $mime2utf8 ."]\n";
```

実行すると….

```
$ perl a.pl
```

```
   org      [表示義務に違反するメールがありましたので、情報提供いたします]
  MIME JIS  [=?ISO-2022-
JP?B?GyRCST08KDVBTDMkSzBjSD8kOSRrJWEhPCVrJCwkIiRqJF4kNyQ/JE4kRyEiPnBKc0RzNiE
kJCQ/JDckXiQ5GyhC?=]
Encode JIS  [=?ISO-2022-
JP?B?GyRCST08KDVBTDMkSzBjSD8kOSRrJWEhPCVrJCwkIiRqJF4kNyQ/GyhC?=
 =?ISO-2022-JP?B?GyRCJE4kRyEiPnBKc0RzNiEkJCQ/JDckXiQ5GyhC?=]
  MIME
UTF-8[=?UTF-8?B?6KGo56S6576p5YuZ44Gr6YGV5Y+N44GZ44KL44Oh4408440r44GM44GC44KK
44G+44GX44Gf44Gu44Gn44CB5oOF5aCx5o+Q5L6b44GE44Gf44GX44G+44GZ?=]
Encode
UTF-8[=?UTF-8?B?6KGo56S6576p5YuZ44Gr6YGV5Y+N44GZ44KL44Oh4408440r44GM44GC44KK
?=
 =?UTF-8?B?44G+44GX44Gf44Gu44Gn44CB5oOF5aCx5o+Q5L6b44GE44Gf44GX44G+44GZ?=]
$ perl a.pl | nkf
   org      [表示義務に違反するメールがありましたので、情報提供いたします]
  MIME JIS  [表示義務に違反するメールがありましたので、情報提供いたします]
Encode JIS  [表示義務に違反するメールがありましたので、情報提供いたします]
  MIME UTF-8[表示義務に違反するメールがありましたので、情報提供いたします]
Encode UTF-8[表示義務に違反するメールがありましたので、情報提供いたします]
$
```

文字化けもしていないようなので、78文字で分割をするencode()を使った方が良いでしょうnkf自身の結果とも一致しているようですUTF-8の結果がちょっと違っているようですけどデコードできるようですから気にスンナ。

```
$ echo "表示義務に違反するメールがありましたので、情報提供いたします"| nkf -Mj
=?ISO-2022-JP?B?GyRCST08KDVBTDMkSzBjSD8kOSRrJWEhPCVrJCwkIiRqJF4kNyQ/GyhC?=
 =?ISO-2022-JP?B?GyRCJE4kRyEiPnBKc0RzNiEkJCQ/JDckXiQ5GyhC?=
$ echo "表示義務に違反するメールがありましたので、情報提供いたします"| nkf -Mj | nkf
表示義務に違反するメールがありましたので、情報提供いたします
$
$ echo "表示義務に違反するメールがありましたので、情報提供いたします" | nkf -Mw
=?UTF-8?B?6KGo56S6576p5YuZ44Gr6YGV5Y+N44GZ44KL44Oh4408440r44GM44GC44KK?=
 =?UTF-8?B?44G+44GX44Gf44Gu44Gn44CB5oOF5aCx5o+Q5L6b44GE44Gf44GX44G+?=
 =?UTF-8?B?44GZ?=
$ echo "表示義務に違反するメールがありましたので、情報提供いたします" | nkf -Mw | nkf
表示義務に違反するメールがありましたので、情報提供いたします
$
```

スクリプトにする理由

スクリプトにする事で、少ない負担で報告(情報提供)の自動化が可能になります。

例えば、筆者の場合bogofilter + Maildrop でSPAMのフィルタリングを行っています。フィルタされたSPAMを先のスクリプトで送るわけです。

現在は

1. SPAM判定したメールを特定のディレクトリへ移動：サーバ

2. 誤ってSPAM判定されたメールをサルベージ：人間
3. 週末までディレクトリに残ったメールをSPAM(不要メール)と判断してアーカイブ：サーバ
4. アーカイブしたメールをコピーし、迷惑メールとして日本データ通信協会へ提供：サーバ

しています。

ほとんど誤判定が無いくらいに学習が進んだbogofilterのおかげで、放置していても問題ないレベルになっています。
※SPAMを消さずにアーカイブしているのは□bogofilterの辞書がおかしくなってしまった時に備えての再学習用です。

とはいっても、やっぱり誤判定は発生するわけで、誤ったメールをSPAMとさせないため人間の目視確認は無くす事ができませんが、報告用メールを手で一つずつ作る作業を行うよりはずっと簡単になります。

筆者にSPAMを送り続ける限り、お上の組織へ報告が続くことになる。 きっとその中の一つくらいは対応してもらえるに違いない、と勝手に思っています。

そういえば最近、迷惑メール中のURLを抽出して、フィルタリングサービスの業者に提供する、って話を聞きました。

業者がメールを送れば送るほど、使えるドメインが減っていく事になります。なかなかうまい手かも。

mail, Perl, MIME-Tools, 技術資料