Net::Telnet

img://www.isogest.org/Home/guest/images/opliti.png
Net::Telnet offre una interfaccia a oggetti per collegarsi a un server su porta TCP, sfruttando in particolar modo il protocollo Telnet.

Ci sono due modi per usarlo: o passo passo, aspettando l'output del server come con un client testuale in prima persona oppure con funzioni automatizzate che svolgono compiti predefiniti.

Ad esempio è possibile collegarsi a un server facendo in automatico il login:

```
use strict;
use Net::Telnet

# Crea l'oggetto Telnet
my $telnet = new Net::Telnet( -host => localhost);

# Eseguo il login
$telnet->login('utente','password');
```

Oppure, se il server non risponde chiedendo nel modo più classico *login* o *username* e *password*, è necessario "attender quanto il server chiede; ad esempio, se fosse necessaria solo la password, è sufficiente il codice seguente:

```
use strict;
use Net::Telnet

# Crea l'oggetto Telnet
my $telnet = new Net::Telnet( -host => localhost);

# Aspetto il testo che già conosco
$telnet->waitfor('/Password:.*/');

# Scrivo la password
$telnet->print('la_password');
```

Una volta instaurata la connessione è possibile mandare dati con vari metodi o eseguire in automatico delle routine qua le impostazioni di telnet venissero modificate dal server.

Come già accennato, il modulo permette collegarsi ad una porta TCP di qualsiasi server anche non Telnet come un ser POP. L'esempio che segue fa proprio questo, visualizzando l'oggetto ed il mittente di ogni messaggio in una casella di p elettronica intterrogandola ciclicamente.

```
#!/usr/bin/perl

use strict;
use Net::Telnet;

# Inizializzazione delle variabili
my ($hostname, $passwd, $username);

$hostname = '<indirizzo>';                # Sostituisci i valori con i tuoi dati
$username = '<username>';
$passwd   = '<password>';

my ($pop, $line, $buffer);

$pop = new Net::Telnet (Telnetmode => 0);  # Creiamo una nuova istanza dell'oggetto
```

```
                                        # Il parametro Telnetmode indica
                                        # se la connessione è di tipo Telnet

$pop->open(Host => $hostname,           # Apriamo la connessione con l'host
           Port => 110);



$line = $pop->getline;                  # Controlliamo la risposta del POP ed
die $line unless $line =~ /^\+OK/;      # interrompiamo il programma a meno che la
                                        # stringa ricevuta inizi con OK

$pop->print("user $username");          # Passiamo all'autenticazione
$line = $pop->getline;                  # inviando username e password e
die $line unless $line =~ /^\+OK/;      # svolgendo gli stessi controlli

$pop->print("pass $passwd");
$line = $pop->getline;
die $line unless $line =~ /^\+OK/;


$pop->print("stat");                     # Controlliamo il totale dei
$line = $pop->getline;                  # messaggi sul server

my ($ok, $numero_messaggi, $bytes_totali) = split /\s/, $line;
die $line unless  $ok =~ /^\+OK/;


# Se nella casella di posta ci sono messaggi controlliamoli uno ad uno
if ($numero_messaggi > 0) {


  # Definiamo la formattazione dell'intestazione e corpo del messaggio
  my ($email, $oggetto, $messaggio);

  printf( "\n%3d messaggi per un totale di %.2f K\n"
        , $numero_messaggi, $bytes_totali / 1024);

  # Per esaminare ogni messaggio, invece di controllare linea per linea evitando
  # così la procedura precedente, sfruttiamo un metodo di Net::Telnet, cmd, che
  # permette di eseguire un comando e redirezionare l'output in una reference
  # precedentemente definita

  $buffer = $pop->buffer;  # Inizializza lo spazio dove il modulo salverà  i dati

  # Inizia il ciclo del controllo dei messaggi
  for ($messaggio = 1;
       $messaggio <= $numero_messaggi;
       $messaggio++) {

    # Esegui il metodo command impostando output alla reference inizializzata.
    # Da notare anche il parametro prompt: una volta eseguito il comando il
    # server pop restituisce un output differente a quella di un normale server
    # telnet o ssh; mediante una espressione regolare è possibile inserire ogni
    # forma di prompt possibile
    $pop->cmd( string => "top $messaggio 0"
             , output => $buffer
             , prompt => "/\n\.\n/"
             , timeout => 20);

    # Con una regular expression estraiamo il risultato
    my $header =  $$buffer;
    $header =~ /From:\s(.+?)\n.+?Subject:\s(.+?)\n/is;

    $email   = $1;
    $oggetto = $2;
```

```
    # Scrivi i risultati
    printf("%03d | %s - %s\n", $messaggio, $email, $oggetto);

    # Svuotiamo l'area di memoria per ricevere altri dati
    $pop->buffer_empty;

  }

# Se non ci sono messaggi
} else {

  print "Nessun messaggio\n";

}

$pop->close;

exit;
```

In Net::Telnet è facile pure il debug: possibile errori durante la stesura del codice possono essere facilmente individuat
salvando su disco l'input e l'output con metodi come [=input_log] ed [=outoput_log].

--
[jontk]

---

commenti di [dada]:

- l'esempio è spropositatamente lungo. lo limiterei a fare login sul server POP e recuperare il numero di messagg
  totali.
- farei un accenno (ma questo può valere anche come nota generale) a com'è "distribuito" il modulo: se la
  documentazione è buona o no, se l'installazione è facile o no, se segue le "direttive CPAN" o no, se richiede un
  compilatore C o no, se i test sono esaustivi o no, etc. etc.

Seppure in ritardo, concordo con i commenti e provvederò a correggere non appena finisco il lavoro che ho in corso.
[jontk]