MasterIT - Scripting Es2

Perl

(Practical Extraction and Report Language)

Sandro.Angius@lnf.infn.it

30/10/2002

Semplice “dos2unix” in Perl

- Editando un file testo su sistemi tipo DOS/WINDOWS viene aggiunto un ^M (Carriage Return), tale carattere puo’ generare errori sui sistemi unix (i quali prevedono solo il ^J cioe’ New Line per segnalare la fine della riga di testo).
  Se non e’ disponibile il comando dos2unix per convertire il file si puo’ usare il seguente mini script Perl:

- **perl -e ‘s/\015//g, print while (<STDIN>)’ < file.dos > file.unix**

coin.pl

```
#!/usr/bin/perl

$testr = 1000000;
$cestr = 0;
$res[0]= 0; # Testa
$res[1]= 0; # Croce

while ($cestr++ < $testr) {

   $ccoin = int(0.5 + rand);
   $res[$ccoin]++;

   printf "%10d  %10d   %10d   %6.3f\n",  $cestr, $res[0], $res[1],100*$res[0]/$cestr   unless ( $cestr % 1000 );

}

exit;
```

Simulazione dell'estrazione di una moneta. Ogni 1000 estrazioni viene riportato il risultato parziale.

crypt.pl

```
#!/usr/bin/perl

printf "Enter password:";
$pwd=<STDIN>;

#srand();

chomp($pwd);

printf("%s ---> <%s>\n", $pwd, crypt($pwd, substr(crypt($pwd, "00"),2 + rand() * 10,2)));

exit;
```

Codifica della stringa in input con il metodo usato da unix per il file /etc/passwd.

makedict.pl

```
#!/usr/bin/perl

while (<STDIN>) {
   $_ = lc $_;
   s/[^a-z]/ /g;
   s/\s+/ /g;
   foreach $word (split(' ')){
      next if ($word =~ /^.$/);
      $dict{$word} = 1;
   }
}

$cnt=0;

foreach $word (sort keys %dict){
   printf("%7d %s\n", ++$cnt, $word);
}
```

Dato un file tramite standard input makedict.pl estrae le parole di piu' di 2 caratteri contenute nel file e le elenca in ordine alfabetico riportandole infine in output.

revsort.pl

```
#!/usr/bin/perl

sub revsort{
   ( (reverse $a) cmp (reverse $b) );
}

while (<STDIN>) {
   $_ = lc $_;
   s/[^a-z]/ /g;
   s/\s+/ /g;
   foreach $word (split(' ')){
      next if ($word =~ /^.$/);
      $dict{$word} = 1;
   }
}

$cnt=0;

foreach $word (sort revsort (keys %dict)){
   printf("%7d %s\n", ++$cnt, $word);
}
```

Come makedict.pl ma ordina le parole dal carattere finale a quello iniziale.

numstr2num.pl

- L'esempio che segue converte i numeri compresi tra 1 e 999.999 scritti in lettere (in lingua italiana) in numeri scritti in cifre, es:

- duecentoventisette ---> 227
- novecentomiladuecentodue ---> 900202

- Per la verifica del corretto funzionamento potrebbe essere usato uno script che fa l'operazione inversa (num2numstr.pl).

numstr2num.pl

```
#!/usr/bin/perl

while ( $strnum = <STDIN> ) {

   chomp($strnum);    $oval= $strnum;    $strnum = lc $strnum;
   if ($strnum =~ /[^a-z]/) {
      print "Input non valido!\n";  next;
   }

   $cnt=1;
   foreach $cifra ("uno", "due", "tre", "quattro", "cinque",
       "sei", "sette", "otto", "nove", "dieci", "undici", "dodici",
       "3dici", "quattordici", "quindici", "sedici", "dicias7",
       "dici8", "dician9") {
      $strnum =~ s/$cifra/$cnt/g;
      $cnt++;
   }
   $cnt=2;    $lchar='i';
   foreach $cifra ("vent", "3nt", "quarant", "cinquant",
                "sessant", "settant", "ottant", "novant") {

      $strnum =~ s/$cifra($lchar|\d)/${cnt}_$1/g;
      $cnt++;
      $lchar='a';
   }
   $strnum=~ s/_(\d)/$1/g;    # <decina>{1,8}
   $strnum=~ s/_.(\d)/$1/g;   # <decina>{2-7,9}
   $strnum=~ s/_./0/g;        # <decina>0

   # {n}<centinaia>{0-99}
   $strnum=~ s/(\d)cento(\d*)/100*$1+$2/eg;

   # 1{0-99}
   $strnum=~ s/cento(\d*)/100+$1/eg;

   # 1{0-999}
   $strnum=~ s/mille(\d*)/1000+$1/eg;

   # <migliaia>{0-999}
   $strnum=~ s/(\d+)mila(\d*)/1000*$1+$2/eg;

   if ($strnum =~ /\D/) {
      print "Conversione non riuscita ($oval ---> $strnum)\n";
   }
   else {
      print "$oval ---> $strnum\n";
   }
}
```

Controllo processi: alarm <time>

```
#!/usr/bin/perl

$timeout = 5;

eval {
   local $SIG{ALRM} = sub { die "alarm\n" }; # NB \n required
   alarm $timeout;
   print "\nPremi invio entro $timeout secondi!!!";
   <STDIN>;
   alarm 0;
};

die if $@ && $@ ne "alarm\n"; # propagate errors

if ($@) {
      # timed out
      print "\n\nTempo scaduto!\n\n";
}
else {
      # didn't
      print "\n\nHai premuto invio in tempo!!!\n\n";
}
exit;
```

Aperture di Socket IP (geturl.pl)

```
use IO::Socket;

$sock = IO::Socket::INET->new(PeerAddr => 'www.lnf.infn.it',
                              PeerPort => 'http(80)',
                              Proto => 'tcp');

die "$!" unless $sock;

$sock->autoflush();

$sock->print("GET www.lnf.infn.it/\015\012");

$document = join('', $sock->getlines());

print "DOC IS:\n$document\n";
```

Un esempio di client/server

(client0.pl)

```
#!/usr/bin/perl -w

use strict;
use IO::Socket;

my ($host, $port, $kidpid, $handle, $line);

unless (@ARGV == 2) { die "usage: $0 host port" }
($host, $port) = @ARGV;

# create a tcp connection to the specified host and port
$handle = IO::Socket::INET->new(Proto => "tcp",
                                PeerAddr => $host,
                                PeerPort => $port)
   or die "can't connect to port $port on $host: $!";

$handle->autoflush(1); # so output gets there right away

print STDERR "[Connected to $host:$port]\n";

# split the program into two processes, identical twins
die "can't fork: $!" unless defined($kidpid = fork());
```

```
# the if{} block runs only in the parent process
if ($kidpid) {
   # copy the socket to standard output
   while (defined ($line = <$handle>)) {
      print STDOUT $line;
   }
   kill("TERM", $kidpid); # send SIGTERM to child
}
# the else{} block runs only in the child process
else {
   # copy standard input to the socket
   while (defined ($line = <STDIN>)) {
      print $handle $line;
   }
}
```

Un esempio di client/server

(server0.pl)

```
#!/usr/bin/perl -w

use IO::File;
use IO::Socket;

sub showfile{
   my($fname, $client) = @_;
   my($fh) = new IO::File;
   my($lines) = 0;
   if (open($fh, $fname)) {
      while (<$fh>) {
         print $client $_;
         $lines++;
      }
      close($fh);
   }
   $lines;
}
$clifile="clients.txt";          $ncon = 0;
$PORT = 9000; # pick something not in use
$server = IO::Socket::INET->new( Proto => "tcp",
    LocalPort => $PORT, Listen => SOMAXCONN, Reuse => 1);

die "can't setup server" unless $server;
```

```
print "[Server $0 accepting clients]\n";
for(;;) {
   $ncon++; print "Waiting connection: $ncon\n";
   $client = $server->accept(); $client->autoflush(1);
   printf "Spawning connection $ncon from %s port %s\n",
           $client->peerhost, $client->peerport;
    if ((fork()) == 0) {  # This is the child process
      print $client "Welcome to $0.\n";
      print $client "Text written by previous clients:\n\n";
      $lines = &showfile($clifile, $client);
      $ftext = new IO::File;
      while ( <$client> ) {
         next unless /\S/; # Skip blank lines
         if (/quit|exit/i) { last; }
         &showfile($clifile, $client), next if (/^showfile$/);
         open($ftext, ">>$clifile");
         printf $ftext "[%5d: %15s ]: %s",
                        $lines++, $client->peerhost, $_;
         close($ftext);
      }
      close $client; print "Connection $ncon closed.\n"; exit;
   }
   else { close $client; } # This is the parent process
}
```

Debugging in Perl

- Switch “-w” *(#!/usr/bin/perl -w)*
  - Conviene utilizzare sempre lo switch *“-w”* perche’di evidenziare alcuni possibili errori come l’uso di variabili non inizializzate o definizioni di variabili mai usate.
- Direttiva “use strict;”
  - La direttiva *“strict”* istruisce Perl ad emettere un errore se si usano variabili con metodi non congrui.
- Perl debugger
  - Il debugger con Perl puo’ essere invocato con *“perl -d <perlscript>”*.

a2p , s2p : un cenno

- E’ possibile convertire uno script *awk* in uno *Perl* tramite il comando *a2p*.

- Per convertire, invece, uno script *sed* in *Perl* si puo’ usare il comando *s2p*.

text.pl

- Dato un file di testo in input contare le parole (di minimo 2 caratteri), i numeri e gli indirizzi email contenuti nel testo.
- Per le sole parole riportare le occorrenze di ogni singola parola e la distribuzione per lunghezza.

Bibliografia

- Perl Programmers Reference Guide
  (una copia si trova in /afs/lnf.infn.it/system/doc/perl/perlman.pdf)

- Copia degli scripts si trova in:
  /afs/lnf.infn.it/project/master.it/doc/Scripting/EsScripts/