Protocollo HTTP, interfaccia CGI e linguaggio Perl

Marco Liverani
liverani@mat.uniroma1.it

5 Dicembre 1996

Sommario

Prima parte: La "tecnologia web"

- Il protocollo TCP/IP, il protocollo HTTP, il "linguaggio" HTML.
- Il server HTTP: cosa è, quali funzioni svolge.
- Il browser come terminale di un sistema client/server.
- L'interfaccia di programmazione CGI: form, script CGI, pagine "dinamiche", passaggio di parametri.

Seconda parte: Il linguaggio Perl

- Caratteristiche fondamentali, istruzioni principali.
- Variabili e tipi: scalari, array e liste, array associativi.
- Espressioni regolari: cosa sono, il pattern matching, il pattern substitution.
- Interazione con altri programmi: alcuni esempi (grep, sort, mail).

Terza parte: Procedure CGI in Perl

- Introduzione, funzioni standard.
- Esempi: archiviazione di dati, ricerca in un archivio.

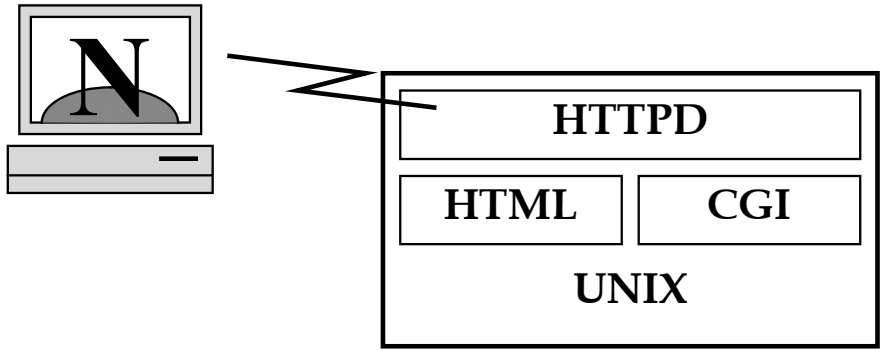
Il protocollo TCP/IP

- È uno dei protocolli di rete più diffusi al mondo; sul TCP/IP si basa la comunicazione fra i nodi della rete Internet.
- Ad ogni nodo della Rete è assegnato un indirizzo IP composto da quattro numeri compresi tra 0 e 254 (es.: 151.100.50.2).
- Mediante opportuni meccanismi (DNS) è possibile convertire gli IP address numerici in indirizzi mnemonici (es.: 141.108.3.218 = magritte.sci.uniroma1.it).
- Sopra al protocollo TCP/IP possono viaggiare informazioni codificate secondo altri protocolli di comunicazione (FTP, Telnet, HTTP).

Il protocollo HTTP

- HTTP = *HyperText Transfer Protocol*.
- Consente l'accesso a documenti ipertestuali in cui vengono realizzati dei link tra file di vario genere (non solo testuali) fisicamente residenti anche su host differenti.
- È gestito da un software (server HTTP) residente sugli host che intendono essere fornitori di informazioni. Chi vuole accedere alle informazioni fornite dal server HTTP deve utilizzare un software client (*browser*) in grado di interpretare le informazioni inviate dal server.
- HTTP è un protocollo "*stateless*": ad ogni richiesta si effettua una connessione al server che viene chiusa al termine del trasferimento dell'oggetto richiesto (pagina HTML, immagine, ecc.).
- HTTP identifica le risorse in rete mediante *URL (Universal Resource Location*):

Il protocollo HTTP




```
$ telnet magritte.sci.uniroma1.it 80
Connected to magritte.sci.uniroma1.it port 80
GET /index.html
Content-type: text/html
<html>
<head><title>Welcome to Magritte</title></head>
<body>
<h1>Magritte, web server del progetto Grafica Avanzata</h1>
...
$ _
```

Il linguaggio HTML

- HTML = *HyperText Markup Language*
- È un semplice "linguaggio" di marcatura del testo, ereditato da SGML (*Standard Generalized Markup Language*).
- Consente di esprimere dei collegamenti ad altri oggetti (pagine HTML, immagini, ecc.) mediante URL.
- Consiste in un insieme di *tag* che consentono di caratterizzare porzioni di testo; è compito del client (*browser*) l'interpretazione dei *tag*.
- Esempio:

```
<HTML>
<HEAD><TITLE>La mia pagina</TITLE></HEAD>
<BODY>
<H1>Benvenuti!</H1>
Questa &egrave; la mia pagina.
Visita <A HREF="http://www.mat.uniroma1.it">Matematica</A>.
</BODY>
</HTML>
```

Il server HTTP

- È un programma che "gira" su un host in attesa di una richiesta di connessione sul suo *socket* (la porta assegnatagli, tipicamente la 80).
- Svolge tre compiti fondamentali:
    1. invia al client le risorse disponibili localmente, richieste mediante l'indicazione di una URL;
    2. richiama eventuali procedure esterne con cui comunica mediante l'interfaccia CGI (*Common Gateway Interface*) per lo scambio di parametri e per ottenere in risposta informazioni in formato HTML;
    3. effettua, ove esplicitamente richiesto dalla sua configurazione, l'autenticazione degli utenti mediante *username* e *password*.
- Il server HTTP svolge quindi un ruolo di interfaccia tra il client remoto che effettua delle richieste sul *socket* ed il sistema che lo ospita, il suo *filesystem* ed altri programmi che girano sulla macchina server.

Il browser

- Il browser è l'applicazione *client* di questo sistema ad architettura "client/server".
- Gira sulla macchina dell'utente remoto, legge ed interpreta l'output in formato HTML.
- Visualizza o gestisce le informazioni codificate in formati a lui noti (es.: immagini GIF o JPEG, filmati QuickTime, scene VRML) e rimanda ad altri programmi esterni presenti sulla macchina client per la gestione di formati non conosciuti (es.: documenti Word, documenti Postscript, ecc.).
- Le procedure CGI **non** vengono eseguite sulla macchina client.
- I programmi in linguaggio Java vengono invece scaricati sul client, "compilati" in tempo reale ed eseguiti su di esso.
- Il browser consente di impaginare l'output indipendentemente dalla piattaforma che lo ospita (X11, Macintosh, Windows, ecc.).

L'interfaccia CGI

- CGI = *Common Gateway Interface*
- È il canale di comunicazione tra il server HTTP (le pagine HTML) e le procedure software che girano sul sistema (procedure e *script CGI*).
- Consente di passare dei parametri ad un programma esterno attraverso una *form* o una semplice URL. Consente anche di catturare l'output della procedura (tipicamente in formato HTML) e di inviarlo al client dell'utente remoto.
- Due sono i metodi per il passaggio dei parametri alla procedura esterna:

  GET: tramite *variabile di ambiente* (`QUERY_STRING`);

  POST: tramite lo *standard input*.

  Il metodo utilizzato viene comunicato alla procedura CGI mediante la variabile di ambiente `REQUEST_METHOD`.

L'interfaccia CGI

- Uno script CGI può essere richiamato mediante una form HTML :

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="/cgi-bin/search.cgi">
Nome: <INPUT NAME=nome> <BR>
Et&agrave;: <INPUT NAME=eta> <BR>
<INPUT TYPE=SUBMIT VALUE="Cerca">
<INPUT TYPE=RESET VALUE="Ripristina">
</FORM>
</BODY>
</HTML>
```

- Oppure mediante una URL diretta (il passaggio dei parametri avviene implicitamente con il metodo GET) :

```
<A HREF="/cgi-bin/search.cgi?nome=Marco&eta=28">
Marco Liverani</A>
```

Caratteristiche fondamentali

- Perl = *Practical Extraction and Report Language*
- È un linguaggio interpretato orientato alla gestione di file ASCII ed alla manipolazione di stringhe mediante *espressioni regolari*.

```
#!/bin/perl
# Primo esempio
$file = shift ;
open (IN, “< $file”) || die “Errore\n\n” ;
@dati = <IN> ;
close(IN) ;
@dati = sort(@dati) ;
foreach $k (@dati) {
   print “$k” ;
}
```

Istruzioni principali

- `if (`*condizione*`) {`
  blocco di istruzioni
  `} else {`
  altro blocco di istruzioni
  `}`

- `while (`*condizione*`) {`
  blocco di istruzioni
  `}`

- `for (`*ass. iniziale* `;` *condizione* `;` *incremento*`) {`
  blocco di istruzioni
  `}`

Variabili e tipi

- Variabili scalari: numeri interi, numeri razionali, stringhe (`$var`) .

  Es. : `$nome = 'Marco' ; $c = $a+$b ; $c = "$a+$b" ;`
  `$riga = <IN> ; $data = `date` ;`

- Liste o array di scalari: `@var` ; gli elementi di una lista sono scalari: `$var[3]`.

  Es. : `@mesi = ('Gen','Feb','Mar') ; $mese[0] = 'Gen' ;`
  `$n = $#mesi ; @nomi = sort @nomi ; @file = <IN> ;`

- Array associativi : `%var` ; gli array associativi sono indicizzati da stringhe :

  Es. : `$cognome{'Marco'} = 'Liverani' ;`
  `@nomi = sort keys %cognome ;`

- Le variabili non devono essere dichiarate: viene fatto un *cast* automatico in base al contesto.

Espressioni regolari

- Consentono di rappresentare un insieme di stringhe di caratteri, specificandone uno schema.
- Sono costruite componendo alcuni simboli che svolgono il ruolo di *meta caratteri* :

  `.` = qualsiasi carattere `\s` = un simbolo di spaziatura

  `\d` = una cifra `\w` = una lettera

  da altri simboli che servono come *quantificatori* :

  `?` = zero o una volta

  `+` = una o più volte

  `*` = zero o più volte

  da *ancore* che forzano la posizione della sottostringa :

  `^` = la stringa inizia per ...

  `$` = la stringa termina per ...

- Altri caratteri rappresentano se stessi : `a`, `b`, `c`, `1`, `2`, `3`, ecc.

Espressioni regolari

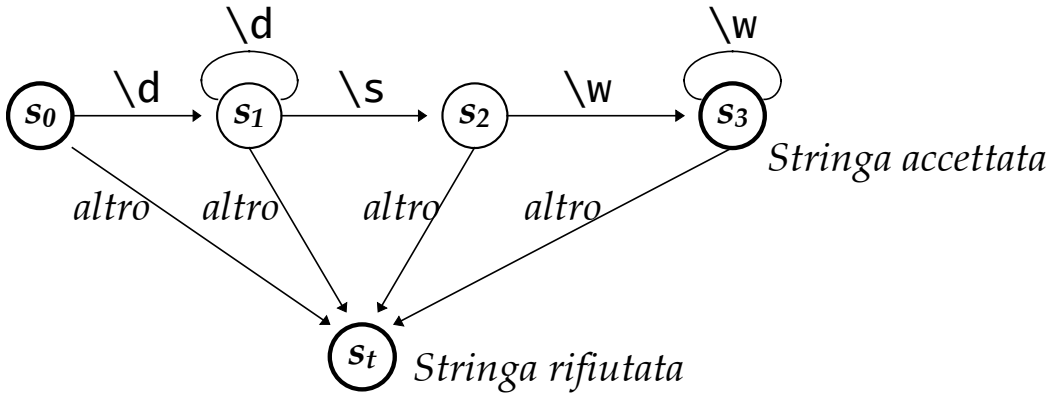
- Esempi :

  `\d+\s\w*` almeno un carattere numerico, uno spazio, un numero arbitrario di caratteri qualsiasi ;

  `^Nome:.+\sEtà:\d+$`

  una riga che inizia con la stringa "`Nome:`", seguita da una stringa non vuota di caratteri qualsiasi, uno spazio, la sottostringa "`Età:`" e che termina con un numero.

- Le espressioni regolari descrivono un *automa riconoscitore*, in grado di accettare o rifiutare determinate stringhe in base alla grammatica specificata mediante l'espressione regolare :



Pattern matching

- Consiste nel verificare che una stringa o una sua sottostringa sia costruita secondo uno schema descritto da una espressione regolare :

  l'istruzione

  `$var =~` */espressione regolare/*`[i]`

  restituisce "vero" se il pattern matching riesce, "falso" altrimenti.

- Esempi :

```
if $testo=~ /^Subject:\s.*$/i {
    print "$testo\n" ;
}

while ($riga = <IN>) {
    $riga =~ /^Subject:\s(.*)$/ && print "$1\n" ;
}
```

Pattern substitution

- Sostituzione di sottostringhe individuate mediante espressioni regolari con altre stringhe.
- L'istruzione

  `$var =~ s/`*espressione regolare/stringa di sostituzione*`/[g][i]`

  sostituisce in `$var` la sottostringa descritta dall'espressione regolare con la stringa di sostituzione.
- Mediante l'uso delle parentesi è possibile memorizzare porzioni di stringa, da richiamare in seguito con `\1`, `\2`, ... e `$1`, `$2`, ...
- Esempi :

```
$var =~ s/Nome:\s(\w+)\sEtà:\s(\d+)/\1, \2 anni/ ;

for ($i=0 ; $i<=$#array ; $i++) {
   $array[$i] =~ s/$v1/$v2/g;
}

$data =~ s/(\d\d)\/(\d\d)\/(\d\d)/\1 $mese[\2] \3/ ;
```

Interazione con programmi esterni

- È possibile richiamare da uno script Perl altri eseguibili esterni in tre modi :
  - ♦ mediante la funzione *system* : `system("cal");`
  - ♦ mediante gli *apici inversi* : `$data = `date`;`
  - ♦ mediante un canale di *pipe* : `open(OUT,"| /bin/mail root");`
- Esempio :

```
$data = `date +%d/%m/%y` ;
@indirizzi = `grep "mat.uniroma1.it" address` ;
foreach $IND (@indirizzi) {
    ($nome, $mail) = split(/:/, $IND) ;
    open(OUT, "| /bin/mail $mail") ;
    print OUT "Roma, $data\n" ;
    print OUT "Caro $nome,\n...\n" ;
    close(OUT) ;
}
```

Uso dell'interfaccia CGI in Perl

- Uno script CGI deve ricevere dei parametri in input attraverso l'interfaccia CGI, effettuare l'elaborazione necessaria e restituire un output formattato secondo un *MIME type* riconosciuto, attraverso il *canale standard di output* che verrà rediretto dall'interfaccia CGI verso il *browser* dell'utente remoto.
- I dati in input sono passati :
  - ♦ mediante la variabile di ambiente `QUERY_STRING` (metodo GET) ;
  - ♦ attraverso lo *standard input* (metodo POST).

  In entrambi i casi `REQUEST_METHOD` indica il metodo utilizzato.
- Esempio :
  `$ENV{REQUEST_METHOD} = "GET"`
  `$ENV{QUERY_STRING} = "nome=Marco+Liverani&citta=Roma"`
- L'output deve essere preceduto da una intestazione che ne dichiara il tipo del contenuto.

  Esempio : "`Content-type: text/html`"

La libreria cgilib.pl

- Esiste una libreria di funzioni del Perl che consente di comunicare facilmente con l'interfaccia CGI. La libreria deve essere richiamata mediante l'istruzione

  ```
  require("/usr/local/lib/perl/cgilib.pl") ;
  ```

- La libreria ci mette a disposizione la funzione "`ReadParse( )`", che restituisce un array associativo (`%in`) in cui vengono inserite tutte le variabili passate attraverso uno qualunque dei due metodi (GET o POST) : `$in{`*variabile*`} =` *valore*.

- Esempio :

  ```
  $in{nome} = "Marco+Liverani" ;
  $in{citta} = "Roma" ;
  ```

Esempio : form di input

- La seguente pagina HTML consente di inserire dei dati e di richiamare lo script Perl che provvederà alla memorizzazione su un file :

```
<HTML>
    <HEAD><TITLE>Input dei dati</TITLE></HEAD>
    <BODY>
        <H1>Inserimento dati</H1>
        <FORM ACTION="/cgi-bin/salva.pl" METHOD=GET>
            Nome : <INPUT NAME=nome SIZE=20><BR>
            Cognome : <INPUT NAME=cognome SIZE=30><BR>
            Citta' : <INPUT NAME=citta SIZE=20><BR>
            <INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
        </FORM>
    </BODY>
</HTML>
```

Esempio : script CGI di archiviazione

- Il seguente script riceve in input i dati della *form* e li archivia su un file :

```
#!/bin/perl
require("/usr/local/lib/perl/cgilib.pl") ;
&ReadParse( ) ;
open(OUT, ">> archivio") ;
print OUT "$in{cognome}|$in{nome}|$in{citta}\n" ;
close(OUT) ;
print <<"END" ;
Content-type: text/html

<HTML><BODY>
Grazie per l'inserimento
</BODY></HTML>
END
```

Esempio : form di interrogazione

- La seguente pagina HTML consente di introdurre dei parametri per effettuare una ricerca sul file archivio :

```
<HTML>
    <HEAD><TITLE>Ricerca</TITLE></HEAD>
    <BODY>
        <H1>Inserisci i parametri per la ricerca in
        archivio</H1>
        <FORM ACTION="/cgi-bin/cerca.pl" METHOD=GET>
            Nome: <INPUT NAME=nome><br>
            <INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
        </FORM>
    </BODY>
</HTML>
```

Esempio : script di ricerca

- Il seguente script effettua una ricerca nel file e visualizza i record individuati :

```
#!/bin/perl
require("/usr/local/lib/perl/cgilib.pl") ;
&ReadParse( ) ;
print "Content-type: text/html\n\n" ;
print "<HTML><BODY>\n" ;
open (ARC, "< archivio") ;
while ($record=<ARC>) {
    ($cognome, $nome, $citta) = split(/\|/, $record) ;
    if ($nome =~ /$in{nome}/) {
        print "Nome: $nome<BR> Cognome: $cognome<BR>
        Citta': $citta<P>\n" ;
    }
}
close(ARC) ;
print "</BODY></HTML>\n" ;
```

Bibliografia

- L. Wall, R.L. Schwartz, *Programming Perl*, O'Reilly & Associates, Inc., Sebastopol, CA, 1990
- B.W. Kernighan, R. Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984 (in italiano : *UNIX*, Zanichelli, Bologna, 1995)
- D. Flanagan, *Java in a Nutshell*, O'Reilly & Associates, Inc., Sebastopol, CA, 1996

- Sul WWW :
  - `http://www.isoc.org` - *Internet Society (ISOC)*
  - `http://www.w3.org/pub/WWW/Protocols/` - *WWW Consortium : link a documenti, articoli e mailing list sui vari protocolli (HTTP, HTML, ecc.).*
  - `http://ds.internic.net/rfc/rfc1945.txt` - *RFC 1945 : Hypertext Transfer Protocol*
  - `ftp://ds.internic.net/rfc/rfc1866.txt` - *RFC 1866 : Hypertext Markup Language*
  - `http://hoohoo.ncsa.uiuc.edu/cgi/` - *The Common Gateway Interface*
  - `http://www.bio.cam.ac.uk/cgi-lib/` - *The cgi-lib.pl home page*
  - `http://www.sandia.gov/sci_compute/html_ref.html` - *HTML Reference Manual*
  - `http://werbach.com/barebones/` - *The Bare Bones Guide to HTML*
  - `http ://www.isinet.it/~marco/perl/` - *Introduzione elementare al Perl*
  - `http://www.mat.uniroma1.it/centro-calcolo/manuali/manuale-unix.ps` - *Introduzione elementare al sistema operativo UNIX per principianti.*