16.4 Un semplice CGI: `hello.exe`

Prima di descrivere e commentare il CGI `hello.exe` vediamo a livello di utente quali sono le azioni che si debbono intraprendere. Per prima cosa invocheremo da browser il CGI `hello.exe` (Figura 16.3): in risposta otterremo la pagina di saluto di Figura 16.4.

Figura 16.3 Invocazione del CGI `hello.exe`

Figura 16.4 Pagina di saluto ottenuta con l'esecuzione del CGI `hello.exe`

Visti gli effetti, vediamo ora quali sono state le "cause" di detti effetti nel Listato 16.1 e commentiamo questo semplice programma CGI.

```
/* hello.c: un esempio di CGI */

/* Includere ciò che serve per lo Standard Input e Output */
#include <stdio.h>

/* Dichiarare il main, come sempre */
int main(int argc, char *argv[])
{
/* Per prima cosa indicare un'informazione necessaria
per l'intestazione della response HTTP                */
     printf("Content-type: text/html\n\n");
/* Inviare su standard Output i tag HTML */
     printf("<head>\n");
     printf("<title>Hello, World</title>\n");
     printf("</head>\n");
     printf("<body>\n");
     printf("<h1>Hello, World</h1>\n");
     printf("</body>\n");

     return 0;
}
```

Listato 16.1 CGI che produce il saluto "Hello World"

Il CGI è un programma lanciato dal server HTTP su richiesta del browser. Come programma dipendente dal server HTTP esso ha uno Standard Input e uno Standard Output che in apparenza funzionano, rispettivamente, come l'inserimento da tastiera e la visualizzazione sul monitor, ma in realtà sono due canali di comunicazione attraverso cui il CGI comunica con il server. Più specificatamente, il server HTTP invia eventuali parametri al CGI tramite STDIN, come vedremo più avanti, e riceve i risultati delle elaborazioni del CGI dallo STDOUT del CGI.

Dunque la prima cosa da fare è quella di includere le funzioni che permettono di gestire lo Standard Input e Output:

```
#include <stdio.h>
```

Dopo questa inclusione il programma procede come sempre per mezzo della dichiarazione della funzione `main`:

```
int main(int argc, char *argv[])
```

A questo punto comincia la costruzione "al volo" della pagina dinamica a partire da un pezzo di informazione relativo all'intestazione della response HTTP:

```
printf("Content-type: text/html\n\n");
                /* si notino le due righe vuote dopo
                   la scritta Content-type: text/html */
```

Come si può osservare, il programma invia su STDOUT, che – ricordiamo ancora una volta – viene intercettato dal server HTTP e poi inoltrato verso il browser, un messaggio che se visualizzato su monitor darebbe luogo alla scritta:

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

Il significato di questo messaggio potrebbe essere così interpretato: "Il contenuto (*content-type*) di quello che sto inviando è un testo ASCII corrispondente alla specifica di una pagina HTML (`text/html`)". Si noti la presenza delle due righe vuote `\n\n`, assolutamente necessarie e, purtroppo, spesso dimenticate.
Infine viene trasmessa, rigo per rigo, la pagina HTML dinamica che realizza il citato messaggio, dapprima inviando su standard output i tag di inizio pagina (in Appendice A riportiamo una breve introduzione al linguaggio HTML che chiarisce il significato dei tag utilizzati nel testo):

```
printf("<head>\n");
printf("<title>Hello, World</title>\n");
printf("</head>\n");
```

poi procedendo con la specifica del corpo della pagina:

```
printf("<body>\n");
printf("<h1>Hello, World</h1>\n");
printf("</body>\n");
```

In effetti la versione statica equivalente di questa pagina dinamica sarebbe stata:

```
<head>
<title>Hello, World</title>
</head>
<body>
<h1>Hello, World</h1>
</body>
```

L'esempio, come abbiamo sottolineato, è molto semplice, ma ci è comunque servito per realizzare il nostro primo programma CGI