Script CGI

Con questo articolo cercherò di spiegare le fasi "standard" di installazione di uno script CGI su un server remoto. Tale argomento è stato trattato in maniera alquanto affrettata nella guida ai CGI proposta in HTML.it, ma viene brevemente spiegato in quasi ognuna delle recensioni degli script CGI più interessanti. Si è scelto di ritornare più approfonditamente sull'argomento per il semplice fatto che, senza le nozioni base per l'installazione di uno script su un server remoto, è inutile scrivere i propri script personalizzati o configurare quelli di altri; inoltre, sappiamo tutti che senza le nozioni di base è molto difficile progredire nella conoscenza di un argomento.
Non faremo qui riferimento, comunque, a nessuno script in particolare, ma cercheremo di mettere in luce tutti gli aspetti salienti della delicata fase dell'installazione e della configurazione.

La prima cosa da prendere in considerazione è sicuramente la locazione dell'interprete Perl installato sul server che ospita le vostre pagine web: se notate, in ogni script (quindi file con estensione .pl o .cgi) la prima riga inizia con un

#!

seguito da una directory (nei sistemi Unix, le directory si indicano con le slash [/], come si usa fare in internet, e non con le backslash [\ ] come succede nei sistemi Windows) che altro non è che la directory che contiene l'eseguibile dell'interprete Perl.
Purtroppo, **la locazione dell'interprete non è standard**, ma varia a seconda del sistema installato sul server e da come lo stesso è stato configurato dall'amministratore.
Alcune locazioni abbastanza standard sono:

#!/usr/bin/perl #!/usr/local/bin/perl

ma non si può mai essere sicuri che una delle due sia quella che fa al caso nostro: per sciogliere questo dubbio basterà collegarci all'home page di chi ospita le nostre pagine e cercare le FAQ (Frequently Asked Questions) dove sarà sicuramente indicata la locazione dell'interprete Perl installato sulle loro macchine.
Per chi sia così fortunato da avere un accesso telnet, invece, basterà leggere l'output del comando:

which perl

lanciato ovviamente dalla shell.

La seconda fase solitamente è quella del settaggio delle variabili.
Intanto: perchè esistono le variabili?
Dal nome, dovrebbe essere chiaro: visto che i sistemi sono differenti gli uni dagli altri, directory, percorsi ed URL non possono essere standard. Inoltre, essendo appunto variabili (inteso qui come aggettivo), permettono un certo grado di configurabilità per lo script, in modo che ognuno possa personalizzare alcuni comportamenti dello script come ritiene più opportuno (mi sto riferendo alle directory che contengono i file dei dati degli script, il modo di visualizzarne l'output e simili).
Solitamente, le variabili sono interne agli script veri e propri, ma in alcuni casi possono essere contenute in altri file, il cui nome potrebbe essere qualcosa del tipo:

nomescript.cfg
config.txt
config
setup.pl - .cgi
nomescriptsetup

e via dicendo: in ogni caso, comunque, questi file hanno un nome che vi fa capire subito che si tratta di file di configurazione. Non ci interessa qui vedere come questi file vengono letti ed interpretati dallo script, sappiate solo che questo è il ruolo che tali file ricoprono. Dove si parla di variabili, inoltre, si parla anche di commenti; spesso infatti per ogni variabile vediamo qualcosa del tipo:

 # Descrizione
$variabile= ....

E' importante sapere che tutte le righe che iniziano con il cancelletto (#) sono chiamate "commenti", ossia vengono tralasciate dall'interprete in fase di compilazione degli script, ma servono solo a chi scrive o legge il codice come "promemoria" o chiarimento. Quindi, le righe sopra ad una variabile che iniziano con il cancelletto altro non sono che una spiegazione del valore della variabile, che è stato introdotto dall'autore presumibilmente per semplificare il procedimento di configurazione-personalizzazione del suo script, e che dovrebbero aiutarci non poco a capire il significato di ogni variabile.

Anche per le variabili non ci sono degli standard: alcune si riferiscono ad URL, altre a percorsi interni, altre ancora a titoli, nomi, colori e quant'altro; ricordiamo quanto detto poche righe sopra e capiremo l'importanza di leggere i commenti introdotti dall'autore.
Ricordiamo ancora un paio di cose che sembrano banalità ma che spesso sono cause di "Error 500"!! Il Perl utilizza dei caratteri speciali e riservati, uno tra tutti la chiocciola (@) per indicare la presenza di un array. Se dovete inserire la chiocciola come valore di una variabile, ricordatevi di inserire davanti a questa anche un backslash che indica all'interprete di interpretare (mi si scusi il gioco di parole) letteralmente il carattere. I caratteri speciali più utilizzati sono: $, ", \, ! (sebbene ve ne siano molti altri).

I percorsi, che nei sistemi Unix si indicano come /percorso/della/directory nei sistemi Windows si indicano come \\percorso\\della\\directory ossia con le backslash precedute da un carattere di escape per il discorso fatto sopra.

Anche se non si parla propriamente di variabili, può accadere di dover modificare, invece che il codice di uno script, il codice di una pagina HTML, ossia l'indice dello script. Molto probabilmente, in queste pagine dovremo andare a modificare dei links che punteranno allo script vero e proprio; sapendo dove lo script sarà caricato, sarà molto semplice far puntare i links allo script stesso, con una piccola attenzione: a volte è necessario passare determinati parametri allo script nel formato http://www.vostroserver.it/cgi-bin/nomescript.cgi?PARAMETRO.
Quindi, nel caso in cui dobbiate modificare un link di tal genere in modo che punti allo script installato sul vostro server, fate grande attenzione a non dimenticare i parametri.

Una volta che le variabili sono impostate (si spera correttamente) si passa al trasferimento dei file: sembra una banalità, ma molto spesso questo passaggio crea non pochi problemi agli utenti. L'upload, sia che venga fatto direttamente dalla linea di comando che tramite qualche programma apposito, **deve sempre essere effettuato in ASCII mode**: il formato ASCII è lo standard per i file di testo, quali ad esempio i file di cui sono formati i nostri script.

Nei programmi dedicati all'FTP, si noterà da qualche parte nella finestra principale (oppure nelle "opzioni" del programma, se proprio l'autore ha voluto complicare le cose) la possibilità di scegliere come i file devono essere caricati: spesso le opzioni tra cui scegliere sono tre: ASCII, binary ed auto. La prima, che è quella che ci serve per le nostre operazioni, è stata brevemente discussa poche righe sopra, la seconda, invece, fa sì che i nostri script non girino sul server, visto che caricare un file di uno script in binary mode equivale alla certezza che questo non possa mai essere eseguito correttamente dal server. La terza opzione lascia molta discrezionalità al programma che stiamo utilizzando visto che (in teoria) esso fa una rapida scansione dei file da caricare e decide come essi devono essere spostati sul server; questa opzione è utile quando si devono caricare in remoto diversi tipi di file, senza per ognuno andare a decidere la modalità dell'upload.
Ma anche questa, che sembra una grande comodità, ha i suoi svantaggi: spesso infatti i programmi si basano sull'estensione dei file, e tramite questa risalgono al loro tipo, senza curarsi troppo di cosa sia effettivamente contenuto all'interno del file stesso. Esistono dei metodi più esatti per un tale tipo di determinazione, che comunque non vedremo in questa sede.
Per chi utilizzi l'FTP direttamente da linea di comando, invece, la modalità ASCII dovrebbe essere la modalità di default: per accertarvene, lanciate il comando "ascii" una volta che siete collegati all'host.

Una volta che i file sono caricati bisognerà impostarne i permessi: non ci soffermeremo qui a parlare di come si impostano i permessi sui file, ci interesseremo piuttosto a prendere in esame alcuni casi particolari:

- **I file con estensione .pl o .cgi**: questi file sono a tutti gli effetti i nostri file eseguibili ma, appena caricati sul server, perdono tale caratteristica (a meno che il vostro host remoto non sia uno di quelli che impostano automaticamente i permessi per gli script da voi caricati). Vista la loro condizione di "inutilità" sul server nelle attuali condizioni, dobbiamo per forza renderli eseguibili con il comando chmod 755 nomefile.

La domanda più ovvia a questo punto è: devo rendere eseguibili tutti i file con estensione .pl o .cgi? La risposta è "sarebbe meglio", tuttavia non è sempre necessario. Supponete di avere uno script che consta di tre file .cgi, script1.cgi, script2.cgi e script3.cgi e che script1.cgi sia il file principale dello script: ovviamente, script1.cgi deve da noi essere reso eseguibile, ma gli altri due? Dipende da come è stato scritto lo script! Nel caso gli altri due script siano chiamati all'interno di script1.cgi con comandi del tipo: *perl script2.cgi*

allora non è necessario che lo script sia eseguibile, in quanto con tale sintassi si chiama in gioco l'interprete che provvede da solo ad eseguire lo script. In altri casi, invece, questo non è possibile, e tutti i tre file devono essere resi eseguibili "a mano". Per chi si chieda come distinguere questi due casi, non disperate, non è necessario leggersi uno ad uno i codici degli script: spesso nei file di aiuto inseriti dall'autore all'interno è ampiamente spiegato quali file necessitano di essere resi eseguibili e quali no. In caso le parole dell'autore siano ambigue o non venga fatta distinzione alcuna, sappiate che rendere eseguibile tutti i file .pl o .cgi non è sbagliato e può togliervi molti problemi.

- Discorso differente va fatto per **gli altri file**: con la premessa che anche per tutti gli altri file dovreste trovare le istruzioni dell'autore, supponiamo di avere a che fare con un counter: lo script traccia tutte le hits ad un determinato sito e, ovviamente, dovrà anche memorizzarle per poterle incrementare di un'unità ogni volta che un nuovo visitatore arriva sulle pagine del sito. Ovviamente, i dati del counter sono immagazzinati in un altro file separato dal counter stesso. Immaginate ora due casi: il file (chiamiamolo data.dat) non sia leggibile o non sia scrivibile.

  Nel primo caso, il counter si lamenterà per il fatto che il file da cui deve leggere quanti visitatori sono già passati per il sito non è leggibile, e quindi, nel caso in cui non fallisca completamente, interpreterà tale dato come uno zero, e segnerà ogni visitatore come fosse il primo.

  Nel secondo caso, invece, riuscirà a leggere il file, ma cosa leggerà mai, se non ha mai potuto scriverci? Ancora, quindi, ogni visitatore sarà il primo (sempre nel caso in cui lo script non fallisca).

  Come regolarsi, in questi casi? Sicuramente se l'autore non dice niente la sua serietà è da mettere in dubbio e spetterà a voi capire quali file devono essere scritti e letti dallo script ed impostarne i pemessi di conseguenza; personalmente, comunque, di script ne ho visti davvero tanti, e solo in un paio di occasioni mancavano le istruzioni su come impostare i permessi. In caso anche voi vi troviate in una simile condizione, fate un atto di generosità ed avvertite l'autore per email.

L'ultima fase è il testing dello script appena installato sul server. Cosa succede se non funziona? O, peggio, adesso che abbiamo installato lo script non sappiamo più dove andare a cercarlo.
E' difficile perdersi tra i link del proprio sito, ma a volte qualche script, se richiamato dal browser come:

http://www.vostroserver.it/cgi-bin/nomescript.cgi

risponde con strani messaggi di errore, comunque differenti dall'odiato Error 500. In questo caso, presumibilmente, bisognerà passare allo script qualche parametro, come:

http://www.vostroserver.it/cgi-bin/nomescript.cgi?admin

per fargli capire, ad esempio, che vogliamo entrare nella schermata riservata all'amministratore dello script stesso. Questo succede negli script che richiedono delle impostazioni preliminari che devono essere fatte direttamente dall'amministratore (presumibilmente voi): senza queste impostazioni, non si può accedere direttamente allo script. Comunque, non preoccupatevi: gli autori non si dimenticano mai di dirvi l'URL alla quale collegarvi per la prima esecuzione, compresi i parametri che dovrete passare allo script.

Discorso leggermente differente va fatto nel caso in cui nell'archivio dello script sia presente un file index.html o nomescript.html: probabilmente, questo file sarà appunto l'indice dello script al quale dovrete collegarvi tramite browser.

Passiamo al secondo caso: lo script si blocca ed esce con un messaggio d'errore.
Credo che pochi abbiano accesso diretto al sistema, potendo così andarsi a leggere i log.

Quindi, dovremo procedere per tentativi:

- La prima cosa da controllare sono i permessi dei file: permesso sbagliato = script bloccato

- Bisognerà poi assicurarsi di aver caricato tutti i file relativi allo script in ASCII mode.

  Se ancora lo script dà problemi, controllate che la prima riga degli script rifletta la reale locazione dell'interprete Perl installato sul server. Se il webserver non trova l'interprete che deve eseguire lo script, il processo di esecuzione si bloccherà.

- Quarta cosa da controllare sarà l'impostazione delle variabili: se avete indicato directory inesistenti, o avete immesso come valore di una variabile il vostro indirizzo email come nome@host.com (che dovrebbe essere invece nome\@host.com), o ancora vi siete dimenticati di inserire qualche backslash davanti ai caratteri riservati o chissà cos'altro ancora, lo script sarà terminato prematuramente.

Il primo caso da prendere in considerazione è che state utilizzando uno script scritto su un sistema diverso: ad esempio se utilizzate uno script adatto a sistemi Unix (vuoi perchè cerca determinati moduli, vuoi perchè è strutturato in modo particolare) su un sistema Windows: in questo caso c'è veramente poco da fare, a meno di riscriversi lo script.

Il secondo caso è che stiate utilizzando uno script scritto in Unix ("in", non "per") su un server Windows: per curiosità andate ad aprire lo script con il blocco note, e cercate di capirne qualcosa. Questo perchè i due sistemi utilizzano degli standard differenti per i "Carriage Return", non contemplati (o almeno non completamente) da sistemi differenti: la soluzione è quella di aprire i file con potenti editor di testo che dovrebbero riuscire ad importare correttamente i file (da qualunque fonte essi vengano) oppure utilizzare degli appositi tools di conversione, come unix2dos o dos2unix. Ovviamente il discorso appena fatto vale sia per script Unix utilizzato sotto Wondows che viceversa. Per farvi un esempio, se da linux eseguo (direttamente da shell) uno script scritto con un editor Dos, ottengo:

Illegal character \015 (carriage return) at script.cgi line 2.
(Maybe you didn't strip carriage returns after a network transfer?)

L'interprete stesso mi parla di qualche errore legato allo "strip" dei ritorni a capo: con un "dos2unix nome_dello_script" converto il formato del file e l'errore sparisce.

L'ultimo doloroso caso è: lo script ha qualcosa che non va, ossia qualche errore di scrittura del codice (rarissimo, ma si trova). Il suggerimento è di eseguire lo script direttamente dalla shell e vedere cosa vi dice l'interprete, correggendo gli errori che vi riporta.

E con questo abbiamo finito di illustrare i passi standard per l'installazione e la configurazione di uno script sul nostro server. Come certamente avrete notato, le cose da ricordare sono molte e devono essere rispettate dalla prima all'ultima per essere sicuri che lo script non si blocchi o dia problemi: non spaventatevi, con un po' di pratica imparerete a gestire un'installazione senza troppi problemi, l'importante è avere la possibilità di eseguire le vostre prove errando e correggendo gli errori. Ed è per questo che il mio consiglio è sempre quello di installarsi in locale un webserver e l'interprete Perl.