“Web Server Apache: architettura, configurazione, principi di funzionamento, amministrazione, aspetti legati alla sicurezza.”

Marco Pagnanini

http://www.marcopagnanini.it

Nella stesura degli argomenti saranno usate le seguenti convenzioni:

- lo standard input e output dei processi ed i comandi saranno indicati in *corsivo*

- Il contenuto dei files, i nomi di files e directories saranno indicati con il font `courier`.

Versione di riferimento Apache: 1.3.28
Distribuzione Linux di riferimento: Slackware 9.1

Indice

1. Breve storia del www e di Apache

Il **www** e' un'architettura hardware e software che permette di accedere a documenti, oppurtunamente formattati, situati in un qualsiasi posto nel mondo: www e' l'acronimo di **World Wide Web,** o "ragnatela mondiale".

La paternita' del www viene fatta risalire ad un fisico del **CERN** (Consiglio Europeo per la Ricerca Nucleare), **Tim Berners Lee**, che voleva ideare un modo semplice e sicuro che permettesse a tutti gli scenziati del mondo di accedere a documenti scientifici.

Ripercorriamo le varie tappe della nascita del www e del **server web** ad oggi piu' diffuso al mondo: **Apache**.

Nel 1989 l'idea di Tim Berners Lee: un sistema basato su **ipertesti** viene proposto al CERN.

Nel 1990 inizia il lavoro per la creazione del primo browser web, nascono i primi siti sperimentali, implementati con server web di poche centinaia di righe in C.

Nel 1993 il CERN distribuisce il proprio server web.
Poi fu la volta del **NCSA** (National Center for Supercomputing Applications) che rilascio' **NCSA Mosaic,** un browser web gratuito.

Nel 1994 NCSA rilascia ancora una volta gratuitamente il proprio server web **httpd**.
Sempre nel 1994 nasce il consorzio www, il **W3C** ad opera del CERN e del **MIT** (Massachusetts Institute of Techologies) **,** che Lee presiede.
Successivamente il codice di httpd 1.3 del NCSA viene modificato con una serie di patch dall'**Apache Group**: nasce cosi' il web server Apache rilasciato sotto licenza **GNU/GPL**.
L'origine del nome e' dovuta al fatto che il primo server web Apache e' nato "patchando" NCSA (a patchy server).
L'Apache Group diviene un consorzio formato da molti programmatori riunitisi liberamente in stile **open source**.

Nel 1995 uno dei programmatori che aveva collaborato alla stesura di Mosaic (**Marc Andreesen**) lascia l'NCSA e fonda una sua societa', la **Netscape**.

Nel 1996 httpd NCSA viene sostituito in tutto il mondo da Apache, inoltre l'architettura software di Apache viene modificata radicalmente in modo da diventare modulare, seguendo la tradizione dei sistemi **UNIX** like. Vengono scritte anche le prime **API**

(Application Programming Interface).

Nel 1997 i server web on line sono piu' di un milione ....

2. Il concetto di Ipertesto

La grande idea di questi documenti del www e' che sono "sfogliabili" (to browse = sfogliare), nel senso che e' come se sfogliassimo un libro, passando da una pagina ad una altra in maniera sequenziale; ma c'e' di piu', e' possibile passare da un argomento ad un altro correlato mediante il concetto di **hypertext** (ipertesto).
Il concetto e' semplice ed alla portata di tutti, anche dei meno esperti: cliccando sui link all'interno di un ipertesto si accede a pagine web logicamente correlate.

Tuttavia il concetto di ipertesto da solo non basta:
e' necessario un metodo di accesso univoco alle risorse nel web (un sistema di denominazione che prende il nome di **URL**), e' necessario un sistema per la formattazione dei documenti accessibili sempre via web (il linguaggio di formattazione o piu' precisamente di marcatura degli ipertesti **HTML**), e' infine necessario un protocollo di comunicazione che permetta al client ed al server di dialogare e di scambiarsi tali documenti (il protocollo **HTTP**).

L'architettura www prevede tutto questo (e altro).

Occorre fare una precisazione perche' spesso si da erroneamente a Lee la paternita' del concetto di hypertext. In effetti tale concetto non era nuovo.
Il termine ipertesto e' stato coniato da **Ted Nelson** nel 1965, ma e' tuttavia grazie a Lee che ha avuto enorme successo, essendo stato inserito nell'architettura del www.

Negli anni a seguire ci fu un grandioso sviluppo del www grazie all'interfaccia intutitiva dei browser web e grazie al loro rilascio gratuito.
L'unico iniziale freno allo sviluppo era costituito dagli alti costi di connessione ad Internet: nel corso degli anni i costi sono andati diminuendo ma rappresentano ancora un freno, soprattutto per le utenze non professionali.

3. Architettura client/server

L'architettura software del www puo' essere divisa in:

- lato client
- lato server

Il lato client prevede un browser, ovvero un software che elabora le pagine inviate dal server web e le mostra a video. Tali pagine sono formattate con uno specifico linguaggio chiamato html e possono includere al loro interno contenuti multimediali, come ad esempio immagini, filmati o suoni. I tipi supportati che possono essere interpretati dai browser web sono gli stessi definiti per la posta elettronica, i cosiddetti tipi **MIME**.
I browser poi si sono evoluti ed hanno permesso non solo di intrepretare i tipi MIME ma anche di interpretare linguaggi di scripting che vengono detti client side come **Javascript**, o addirittura includono la cosiddetta **Java Virtual Machine** (**JVM**) che permette di eseguire con il browser programmi "sicuri" scritti nel linguaggio di programmazione **Java**.
Il protocollo che permette di effettuare la richiesta di trasferimento di un documento formattato in html da e per il server web e' il protocollo **HTTP** (Hyper Text Transfer Protocol). HTTP permette di accedere ai documenti specificando un **URI** (Uniform Resource Identifier) o piu' in particolare un URL (Uniform Resource Locator), la distinzione e' ancora puramente accademica dato che l'URI e' ancora oggetto di ricerca.
I due metodi piu' utilizzati nel protocollo HTTP sono il metodo **GET** ed il metodo **POST**.
In generale GET serve per recuperare informazioni dal server mentre POST per inviare informazioni al server.

Dal lato server la cosa e' piu' complicata perche' deve essere presente un qualche software che attende le richieste ed in base alle stesse invia delle risposte.
Nella grande ragnatela del www ci sono molti server web che "girano" sotto forma di **demoni** in background ed attendono le richieste di connessione sulla porta 80 **TCP** .
Un client che volesse collegarsi con un server web effettua una richiesta del tipo:

http://www.sito.dominio.com:80/
GET / HTTP/1.1

Il **Fully Qualified Domain Name** (www.sito.dominio.com) sara' poi "mappato" su un indirizzo **IP** in qualche **DNS** (Domain Name Server); grazie ai protocolli di rete TCP/IP si potra' raggiungere il server che nel frattempo e' in ascolto. Non appena ricevuta la

richiesta il server effettuera' la ricerca del documento richiesto (il metodo GET del

protocollo HTTP nell'esempio sopra richiede il documento 'root'), se tale documento

sara' presente verra' inviato al client (codice 200 OK), altrimenti sara' generato un messaggio di errore con codice compreso tra 500 e 599 (errori del server), o il ben noto codice 404 (Not found).

Le informazioni che il server invia al client posso essere inviate "come sono" o possono essere interpretate dal server stesso (programmi **CGI**, script server side in **Perl** o **PHP**), o addirittura cifrate grazie al protocollo **SSL** (Secure Socket Layer).
Il client che si collega al server puo' essere un browser web o un qualunque altro applicativo che possa "dialogare" su una rete TCP/IP come **wget** o **telnet**.

Nel protocollo HTTP 1.1, rispetto alla versione 1.0, e' stato aggiunto un header che descrive i dati trasferiti in modo che il browser si comporti di conseguenza. Il campo dello header che e' piu' utilizzato e' sicuramente quello che prevede che il testo sia formattato in HTML:
Content-Type: text/html
questo header ad sempio compare all'interno del tag <meta> contenuto nello header dalla pagina HTML.

Un'altra miglioria apportata con il protocollo HTTP 1.1 e' l'inserimento nello header di un campo che specifica il FQDN al quale si desidera accedere, in luogo del solo indirizzo IP: questo ha permesso di utilizzare il cosiddetto **Virtual Hosting** che consente, a partire da un unico indirizzo IP, e dunque da una singola macchina, di generare opportune risposte da parte del server web in base al nome host.
In definitiva a partire da una singola macchina, grazie al Virtual Hosting, e' possibile gestire piu' siti web, ognuno corrispondente ad uno specifico nome host.

4. Architettura di Apache

Nei sistemi **GNU/Linux** Apache viene avviato sotto forma di demone in ascolto su una determinata porta TCP, generalmente la porta 80.
Se si ha Apache avviato sulla propria macchina molto probabilmente il comando **top** o il comando **ps** vi mostreranno la presenza di diversi processi con il nome **httpd**.

Se non vi sono richieste di connessione i processi httpd sono in attesa (**idle**).
Non appena arrivano le richieste di connessione di norma non vengono generati nuovi processi autonomi, sono quelli in attesa che eventualmente mediante la system call

fork ne generano di nuovi.
Questo meccanismo, chiamato **preforking**, permette una gestione molto piu' veloce delle connessioni, attraverso una gerarchia di processi.
Il server principale, ovvero il programma demone principale, avvia una serie di processi per gestire diciamo N richieste dei client remoti; se le richieste dovesser o essere in numero maggiore di N allora i vari processi a loro volta eseguono la **syscall** fork per generare altri processi, in una gerarchia ad albero.

Questo modello e' molto piu' affidabile e robusto di un singolo processo che risponde a tutte le richiste da solo, anche se una gestione per mezzo di **thread** in luogo dei singoli processi risulta essere piu' performante, soprattutto perche' i thread richiedono molte meno risorse e sono molto piu' veloci dei processi (nelle versioni 2.x e' avvenuto il definitivo passaggio ad una architettura multithreaded).

Apache si divide in **core** e **moduli**, similmente a cio' che avviene per il kernel di Linux: una entita' statica, compilata ed eseguita che "gira" sempre sulla macchina, tanti moduli aggiuntivi (anche essi compilati) che possono essere caricati e scaricati in modo scalabile: tali moduli permettono di aggiungere nuove funzionalita' che il core non mette a disposizione.

Ad esempio il supporto per **PHP** (Hypertext Preprocessor) e' reso possibile grazie alla presenza del modulo chiamato **mod_php**.

In effetti in tutti i moderni sistemi operativi derivati da **UNIX** esiste il meccanismo chiamato abitualmente **DSO** (Dynamic Shared Objects) che permette il collegamento (linking) e caricamento (loading) di moduli che forniscono funzionalita' aggiuntive anche a **run-time:** file eseguibili possono ad esempio essere caricati nello spazio degli indirizzi di una applicazione che e' gia in esecuzione.

Questi DSO sono raccolti nelle cosiddette **shared libraries,** sono file che terminano con l'estensione `.so`, contenuti nella directory `/usr/lib` e `/usr/libexec` .

5. Nuove funzionalita' nella versione 2.0

Nella nuova versione 2.0 le modifiche architetturali prevedono miglioramenti sia al cosiddetto core che ai moduli di Apache.

Ecco un elenco delle principali modifiche:

Modifiche al core di Apache

Unix Threading:
Supporto per i thread nei sistemi che seguono lo standard POSIX.

Miglior supporto per architetture non *nix:
Apache e' ora piu' veloce e piu' stabile in architetture come BeOS, OS/2 e Windows, anche se si continuano a registrare le migliori prestazioni nei sistemi *nix-like.

Nuove API di Apache:
Le API sono state riscritte e sono state aggiunte nuove funzioni che evitano di aggiungere eventuali patch ad Apache per averle disponibili.

Supporto Ipv6:
Supporto per il nuovo protocollo IPv6.

Filtri:
I moduli di Apache sono stati riscritti per far si che essi si comportino come filtri, questo puo' ad esempio permettere che l'output di altri script (come gli script CGI) possa essere analizzato e filtrato, anche al fine di convertire i file "al volo".

Supporto multilingua:
I messaggi da parte del server ai client (browser ad esempio) sono ora forniti in varie lingue, utilizzando vari set di caratteri.

Configurazione semplificata:
Molte direttive che generavano confusione sono state semplificate. Ad esempio le direttive *Port* e *BindAddress* sono state eliminate. Al loro posto si utilizza la sola direttiva *Listen*.

Aggiornamento della libreria delle espressioni regolari:
Apache 2.0 include ora la "Perl Compatible Regular Expression Library" (PCRE). Le espressioni regolari usano ora la sintassi della versione 5 del linguaggio di scripting Perl.

Alcune delle modifiche ai moduli di Apache

mod_ssl:

Questo modulo e' stato riscritto e migliorato: si tratta del modulo che permette di interfacciarsi con i protocolli di cifratura SSL/TLS forniti da **OpenSSL**.

mod_deflate:

Questo modulo permette di inviare i dati in formato compresso ai browser che ne facciano richiesta.

mod_charset_lite

Nuovo modulo sperimentale che permette di effettuare la traduzione tra i vari set di caratteri.

6. Compilazione ed installazione

Ci sono due possibilita' di installazione:
installazione dei file binari per mezzo di pacchetti precompilati o compilazione dei sorgenti e installazione degli eseguibili.

La prima strada e' consigliabile a coloro che non hanno grosse esigenze ed installano Apache ad esempio per scrivere qualche script in php.

La seconda strada e' preferibile e quasi obbligatoria nel caso in cui il web server Apache debba gestire molte richieste di connessione, come nel caso dei provider: infatti la compilazione da vita a file eseguibili specifici per la macchina che permettono di raggiungere maggiori prestazioni.

I pacchetti precompilati di norma sono installati di default in tutte le maggiori distribuzioni come Slackware, Debian o Red Hat.

Download

Prima di tutto occorre scaricare dal sito http://www.apache.org i sorgenti ed il file delle checksum (o message digest):

$ wget http://www.apache.org/dist/httpd/apache_1.3.28.tar.gz
$ wget http://www.apache.org/dist/httpd/apache_1.3.28.tar.gz.md5

Controlliamo l'integrita' del file appena scaricato:

$ md5sum -check apache_1.3.28.tar.gz.md5

Se il file risulta essere integro procediamo con la decompressione e dearchiviazione del **tarball:**

$ tar xzvf apache_1.3.28.tar.gz

Troveremo a questo punto una directory col nome di `apache_1.3.28`

Configurazione (creazione dei Makefiles)

Il processo di configurazione che precede quello di compilazione che analizzeremo e' stato introdotto a partire dalla versione 1.3 di Apache e prende il nome di **APACI** (Apache AutoConf-style Interface)**.** APACI ha il compito di creare i files che contengono le specifiche di come deve avvenire la compilazione.

Il cuore di APACI e' lo script `configure` che si trova nella directory principale del sorgente di Apache. Di solito `configure` viene avviato con una serie di opzioni per personalizzare la propria versione di Apache.

Generalmente si utilizzano le seguenti opzioni:

```
$ ./configure –prefix=/etc/apache \
> --enable-module=most \
> --disable-module=mod_auth_dbm \
> --enable-shared=max
```

--prefix permette di cambiare la directory di installazione, *--enable-module=most* attiva tutti i moduli standard che sono utilizzabili su tutte le piattaforme supportate da Apache.

Se avessimo inserito invece *–enable-module=all* sarebbero stati attivati tutti i moduli standard.
--disable-module disabilita uno specifico modulo, nel nostro caso si tratta di *mod_auth_dbm.*

Lo script `configure` crea un insieme di istruzioni, contenute nei cosiddetti **Makefiles**, che guidano il processo di compilazione, effettuato dal comando **make**, in base alle opzioni che abiamo visto sopra ma anche in base alle librerie disonibili nel sistema o ad altre caratteristiche del sistema utilizzato.

Compilazione e installazione

Dopo la fase di configurazione, se non ci sono stati errori, come abbiamo visto, vengono generati una serie di Makefiles: a questo punto e' il momento di lanciare il comando make:

$ make

Il processo di compilazione potrebbe durare qualche minuto.

Non appena terminato passiamo ad utente root e installiamo:

$ make install

Se non ci sono errori possiamo passare alla fase di avvio del server.

7. Avviare il server

Il passo successivo e' quello di avviare il demone di Apache, tutti i comandi che seguono devono essere lanciati come utente root.

Avvio manuale

Per tutte le distribuzioni esiste la possibilita' di usare il comando **apachectl** in questo modo:

 # apachectl start

Per conoscere le altre possibilita' di utilizzo di *apachectl* avviatelo senza opzioni.

In base alle diverse distribuzioni diverso sara' la possibilita' di lanciare Apache in modo manuale (senza l'utilizzo di apachectl), anche perche' diverso e' il nome dell'eseguibile: in slackware l'eseguibile si chiama "httpd" mentre in Debian si chiama "apache".

Avvio automatico al boot

Questa e' la modalita' sicuramente piu' utilizzata.
Nel caso di Slackware, e dei sistemi *__BSD__ in generale, apache viene lanciato al boot da uno script apposito, si tratta di : */etc/rc.d/rc.httpd* .

Per attivare e disattivare l'avvio di Apache in fase di boot e' possibile attivare o disattivare il bit di esecuzione dello stesso script, in questo modo,
per attivare:

 # chmod +x /etc/rc.d/rc.httpd

per disattivare:

 # chmod -x /etc/rc.d/rc.httpd

Nel caso di Debian viene invece utilizzato il meccanismo di inizializzazione tipico dei sistemi **System V**: anche in Debian Apache viene lanciato al boot da uno script apposito, */etc/init.d/apache*, la differenza e' che per attivare e disattivare Apache al boot occore eliminare o aggiungere il link allo script in determinate directories in /etc/rc*.d/ dove * indica il runlevel (1,2,3,4,5,6,S) .

Per avere maggiori informazioni riguardo i runlevels e' possibile consultare Linux Run Levels di Riccardo Balestieri oppure gli Appunti di informatica libera di Daniele Giacomini alla voce "Procedura di inizializzazione del sistema (System V)".

Per la gestione dei runlevel esiste anche un comodo script sia in Debian che in Gentoo, si tratta rispettivamente di **update-rc.d** e di **rc-update**.

A questo punto sarebbe bene controllare che il demone sia stato effettivamente avviato mediante il comando:

 # ps aux | grep httpd

```
root     6425  0.0  1.7 74856 4556 ?  S  16:15  0:00 /usr/sbin/httpd
nobody   7069  0.0  1.9 75000 4908 ?  S  19:22  0:00 /usr/sbin/httpd
nobody   7070  0.0  1.9 75000 4908 ?  S  19:22  0:00 /usr/sbin/httpd
nobody   7071  0.0  1.7 74856 4564 ?  S  19:22  0:00 /usr/sbin/httpd
nobody   7074  0.0  1.7 74856 4564 ?  S  19:22  0:00 /usr/sbin/httpd
nobody   7075  0.0  1.7 74856 4564 ?  S  19:22  0:00 /usr/sbin/httpd
nobody   7086  0.0  1.7 74856 4564 ?  S  19:25  0:00 /usr/sbin/httpd
```

(sostituire **httpd** con **apache** nel caso di Debian).

Dall'output e' possibile vedere che c'e' un solo processo padre server (quello dell'utente root) e 6 processi figli, tutti in attesa (S=sleep).

I processi server figli sono quelli che realmente gestiscono le richieste di connessione, dunque se le connessioni saranno in numero maggiore di 5 e simultanee, il processo padre dovra' chiamare la fork per creare altri processi figli al fine di gestire le richieste pendenti.

Abbiamo avviato Apache, adesso lo dobbiamo configurare a dovere affinche' possa soddisfare al meglio le nostre esigenze.

8. Esempio di accesso manuale al web server Apache

Faremo ora un esempio di accesso al server web Apache per mezzo del comando **telnet** che ci permettera' di collegarci alla porta TCP 80 in cui e' in ascolto il server web Apache.

Il comando e' il seguente:

$ telnet linux2 80

'linux2' e' il nome host della mia macchina, la porta a cui voglio collegarmi invece e' la 80.

L'output del comando e' il seguente:

Trying 192.168.1.2...
Connected to linux2.
Escape character is '^]'

A questo punto la connessione e' avvenuta, il socket TCP e' stato creato, il web server Apache e' in attesa di comandi.

Supponiamo di voler scaricare la pagina di default del cosiddetto **DocumentRoot** di Apache (vedremo poi di cosa si tratta), mediante il seguente comando:

GET / HTTP/1.1

GET e' il metodo che permette di ricevere la pagina web, il protocollo specificato e' il

piu' recente HTTP/1.1, come output avremo ad esempio la pagina index.html che risiede nel server:

<html>
<head>
<title>titolo pagina index.html</title>
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
</head>
<body>
corpo della pagina
</body>
</html>

Se ad esempio avessimo chiesto la pagina index2.html:

GET /index2.html HTTP/1.1

avremmo avuto il seguente messaggio di errore in quanto la pagina non e' presente nella DocumentRoot del server:

<html>
<head>
<title>404 NOT FOUND</title>
<H1>Not Found</H1>
The requested URL /index2.html was not found on this server.<P>
<HR>
<ADDRESSApache/1.3.27 Server at linux2.mydomain Port 80</ADDRESS>
</body>
</html>

Un altro semplice esempio

Come fare per vedere su che tipo di server web si trova un sito?

$ telnet www.marcopagnanini.it 80

Utilizziamo il metodo HEAD in questo modo, al fine di scaricare solo l'header della pagina e non il body:

HEAD / HTTP/1.1

otterremo il seguente output che ci permette di avere alcune informazioni sul server web remoto:

HTTP/1.1 400 Bad Request
Date: Fri, 19 Sep 2003 18:01:42 GMT
Server: Apache/2.0.46 (Unix) PHP/4.3.2
Connection: close
Content-Type: text/html; charset=iso-8859-1

9. Configurazione: httpd.conf (versione di riferimento: 1.3.28)

Verra' ora presentato il file di configurazione di Apache, **httpd.conf** (in alcune distribuzioni puo' avere nomi diversi o puo' addirittura essere diviso in piu' file). Innanzitutto per sapere come si chiama e dove si trova il vostro file di configurazione di Apache, e per avere altre importanti informazioni, provate ad avviare questo comando come utente root:

$ httpd -V

dovreste avere un output del tipo:

Server version: Apache/1.3.27 (Unix)
Server built: Mar 5 2003 13:50:46
Server's Module Magic Number: 19990320:13
Server compiled with....
-D EAPI
-D HAVE_MMAP
-D HAVE_SHMGET
-D USE_SHMGET_SCOREBOARD
-D USE_MMAP_FILES
-D HAVE_FCNTL_SERIALIZED_ACCEPT
-D HAVE_SYSVSEM_SERIALIZED_ACCEPT
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D HARD_SERVER_LIMIT=256
-D HTTPD_ROOT="/usr"
-D SUEXEC_BIN="/usr/sbin/suexec"
-D DEFAULT_PIDLOG="/var/run/httpd.pid"
-D DEFAULT_SCOREBOARD="/var/run/httpd.scoreboard"
-D DEFAULT_LOCKFILE="/var/run/httpd.lock"
-D DEFAULT_ERRORLOG="/var/log/apache/error_log"
-D TYPES_CONFIG_FILE="/etc/apache/mime.types"
-D SERVER_CONFIG_FILE="/etc/apache/httpd.conf"
-D ACCESS_CONFIG_FILE="/etc/apache/access.conf"
-D RESOURCE_CONFIG_FILE="/etc/apache/srm.conf"

SERVER_CONFIG_FILE="/etc/apache/httpd.conf" indica il file di configurazione di Apache, quello che sara' preso in esame tra poco.

Apache viene configurato impartendogli delle **direttive**, grazie ad esse e' possibile istruire Apache su come deve comportarsi.

Le direttive operano in determinati contesti:

- **General Context**
- **Container Context**
- **Virtual Host Context**
- **.htaccess Context**

Le direttive che operano nel contesto Generale si applicano globalmente alla configurazione del server, operano cioe' in un contesto globale.

Il contesto Container o Contenitore e' quello compreso tra le direttive contenitore <Directory>, <Files> e <Location>, le direttive all'interno dei contenitori si applicano sono al contesto Contenitore stesso (alle direttive contenitore si deve applicare il tag di chiusura in modo simile al linguaggio HTML).

Il contesto Virtual Host e' quello contenuto all'interno della direttiva contenitore <VirtualHost>, il contesto e' considerato in modo separato perche' gli host virtuali operano separatamente, ignorando le impostazioni globali del server, essendo infatti loro stessi dei server agli occhi dei client che si collegano.

All'interno del file `.htaccess` sono contenute le direttive che si applicano al contesto della directory in cui il file stesso si trova, allo stesso modo di cio' che accade per le direttive contenitore <Directory>. Tali direttive possono essere disabilitate tramite la direttiva **AllowOverride** applicata alla directory in cui `.htaccess` si trova.

Le direttive di configurazione sono raggruppate in tre sezioni principali:

1. Direttive che controllano l'ambiente globale di Apache
2. Direttive che definiscono i parametri del server principale (main server configuration), ovvero quelle direttive che devono gestire le richieste che non sono dirette agli host virtuali (ma appunto al server principale). Queste direttive definiscono anche i valori di default di tutti gli host virtuali.
3. Parametri di configurazione del virtual hosting, tecnica che permette al web server Apache di gestire le diverse richieste che arrivano via web relative a differenti indirizzi IP o nomi di host, cosa resa possibile anche grazie al protocollo HTTP versione 1.1 .

Le direttive possono essere ulteriormente suddivise in direttive core, ovvero quelle built-in nell'eseguibile di Apache (in fase di compilazione), e direttive dei moduli, ovvero quelle messe a disposizione dai moduli aggiuntivi di Apache.

Per spiegare le direttive principali di configurazione utilizzero' direttamente un file di configurazione funzionante:

```
## httpd.conf -- Apache HTTP server configuration file
#
# Queste righe non sono altro che commenti, come tali vengono
# ignorati
#

### Sezione 1: ambiente globale di Apache
##
# ServerType: Indica se Apache debba essere avviato come demone
# o come processo autonomo (standalone).
# Nel caso di processo demone lo spawn (l'avvio) e' fatto da
# inetd o xinetd
#
ServerType standalone

#
# ServerRoot: directory root del server, tutti i percorsi sono
# relativi a ServerRoot
#
ServerRoot "/usr"

#
# LockFile: possiamo lasciare il file di lock di default
# (lasciando il commento), a meno di particolari esigenze
#
#LockFile /var/run/httpd.lock

#
# PidFile: file dove viene registrato il process id
#
PidFile /var/run/httpd.pid

#
# ScoreBoardFile: file in cui vengono memorizzate informazioni
# interne al processo server, per la comunicazione tra processi
# padre e figlio.
# Non e' richiesto da tutte le architetture, ma solo dai sistemi
# che non supportano la memoria condivisa.
# Non e' il caso di Linux, quindi la riga e' commentata
#
#ScoreBoardFile /var/run/httpd.scoreboard
```

```
#
# I due file contenevano la maggior parte delle direttive nelle
# prime versioni di Apache: oggi per motivi di chiarezza tutte
# le direttive sono contenute in questo unico file (httpd.conf)
#
#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

#
# Timeout: imposta il timeout del server
#
Timeout 300

#
# KeepAlive: abilita le connessioni persistenti (piu' di una
# richiesta per connessione)
#
KeepAlive On

#
# MaxKeepAliveRequests: numero massimo di richieste da abilitare
# durante una connessione persistente. 0 per richieste
# illimitate.
# Usare un numero alto per ottenere le massime prestazioni.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: imposta il timeout per le richieste
# successive nelle connessioni persistenti
#
KeepAliveTimeout 15

#
# MinSpareServers, MaxSpareServers: rispettivamente numero
# minimo e massimo di processi server figlio inattivi che
# devono essere presenti
#
MinSpareServers 5
MaxSpareServers 10

#
# StartServers: imposta il numero di processi figlio creati
# inizialmente all'avvio, quelli che avranno username nobody.
#
```

```
StartServers 5

#
# MaxClients: numero massimo di processi figlio che possono
# essere avviati, dunque massimo numero di richieste che
# possono essere elaborate "contemporaneamente"
#
MaxClients 150

#
# MaxRequestsPerChild: numero massimo di richieste che un
# processo server figlio deve gestire prima che muoia
# spontaneamente, 0 per richieste illimitate
#
# (300000 su server Cobalt)
#
MaxRequestsPerChild 300000

#
# Listen: Permette di collegare (bind) Apache a indirizzi IP
# specifici o porte. Vedi anche la direttiva <VirtualHost>
# Naturalmente si possono usare IP pubblici o privati
#Listen 3000
Listen 192.168.100.25:8080
Listen 192.168.1.2:80

#
# BindAddress: indirizzo dell'interfaccia di rete sul quale
# il server rimane in ascolto delle connessioni, puo' anche
# essere specificato un nome di host.
# Per evitare confusione viene utilizzata la sola direttiva
# Listen.
#
#BindAddress 192.168.1.2

#
# Dynamic Shared Object (DSO) Support, supporto per gli Oggetti
# Dinamici Condivisi
#
# Per poter utilizzare le funzionalita' dei moduli come DSO
# bisogna inserire la corrispondente linea `LoadModule' in modo
# che le direttive contenute nel modulo possano essere
# disponibili prima di essere utilizzate: httpd -l per una
# lista dei moduli statici o built-in, cioe' moduli inclusi
# staticamente nel core in fase di compilazione.
# Non cambiare l'ordine dei moduli !
```

```
#
# LoadModule: carica il modulo DSO e fa il link nel file oggetto
# o nella libreria
#
LoadModule vhost_alias_module libexec/mod_vhost_alias.so
LoadModule env_module         libexec/mod_env.so
LoadModule define_module      libexec/mod_define.so
LoadModule config_log_module  libexec/mod_log_config.so
LoadModule mime_magic_module  libexec/mod_mime_magic.so
LoadModule mime_module        libexec/mod_mime.so
LoadModule negotiation_module libexec/mod_negotiation.so
LoadModule status_module      libexec/mod_status.so
LoadModule info_module        libexec/mod_info.so
LoadModule includes_module    libexec/mod_include.so
LoadModule autoindex_module   libexec/mod_autoindex.so
LoadModule dir_module         libexec/mod_dir.so
LoadModule cgi_module         libexec/mod_cgi.so
LoadModule asis_module        libexec/mod_asis.so
LoadModule imap_module        libexec/mod_imap.so
LoadModule action_module      libexec/mod_actions.so
LoadModule speling_module     libexec/mod_speling.so
LoadModule userdir_module     libexec/mod_userdir.so
LoadModule alias_module       libexec/mod_alias.so
LoadModule rewrite_module     libexec/mod_rewrite.so
LoadModule access_module      libexec/mod_access.so
LoadModule auth_module        libexec/mod_auth.so
LoadModule anon_auth_module   libexec/mod_auth_anon.so
LoadModule dbm_auth_module    libexec/mod_auth_dbm.so
LoadModule digest_module      libexec/mod_digest.so
LoadModule proxy_module       libexec/libproxy.so
LoadModule cern_meta_module   libexec/mod_cern_meta.so
LoadModule expires_module     libexec/mod_expires.so
LoadModule headers_module     libexec/mod_headers.so
LoadModule usertrack_module   libexec/mod_usertrack.so
LoadModule unique_id_module   libexec/mod_unique_id.so
LoadModule setenvif_module    libexec/mod_setenvif.so

#
# ExtendedStatus: controlla la memorizzazione di statistiche
# per la visualizzazione della pagina di stato: on informazioni
# dettagliate (diminuzione prestazioni), off informazioni
# minimali.
# E' possibile le informazioni all'URL:
# http://nomehost.tld/server-status?refresh=5
#
ExtendedStatus On
```

```
### Sezione 2: Configurazione principale del server (Main
# Configuration)
#
# Tutte le direttive contenute nel container <VirtualHost>
# scavalcheranno (will override) le direttive contenute in
# questa sezione
#

#
# Se la direttiva ServerType e' stata definita per essere
# avviata in modalita' autonoma (standalone), ovvero avviata
# ad esempio da inetd, le prossime direttive non avranno
# effetto in quanto le stesse configurazioni saranno definite
# da inetd stesso.
#

#
# Port: La porta in cui il server in modalita' standalone si pone
# in ascolto. Se la porta e' <= 1023 (well known ports)
# occorrera' avviare httpd da utente root
#
Port 80

#
# Se vuoi lanciare httpd da utente non root dovrai inizialmente
# avviarlo da root per poi cambiare grazie alle due direttive
# sotto
#
User nobody
Group nobody

#
# ServerAdmin: l'indirizzo email a cui arrivera' il feedback di
# chi ha avuto problemi con il server: questo indirizzo sara'
# infatti contenuto nelle segnalazioni di errore che il server
# inviera' ai client
#
ServerAdmin info@marcopagnanini.it
#

#
# ServerName: indica il FQDN da usare per la creazione di URL di
# redirezione, deve essere un nome DNS valido.
# Eventualmente si puo' anche inserire l'indirizzo IP
```

```
#
ServerName linux2.mydomain
#

# DocumentRoot: directory principale dei documenti web
#
DocumentRoot "/var/www/htdocs"

#
# Da questo punto in poi fino alla fine della Sezione 2
# puo' essere configurata ogni directory a cui Apache ha
# accesso,  ovvero ogni directory che discende dalla
# DocumentRoot.
# Le righe successive scavalcano (overrides) le righe precedenti
#
# Dapprima viene fissata una regola di default,
# una "policy" che sia restrittiva: per i documenti in root (/)
# si decide di permettere di seguire i link simbolici, ma non
# si permette a eventuali file .htaccess di scavalcare le
# direttive. Se cosi' non fosse chiunque abbia i permessi di
# scrittura potrebbe inserire i file .htaccess a suo
# piacimento, e questo porterebbe a problemi di sicurezza
# evidenti
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Da questo punto in poi bisogna specificare tutte le direttive
# per ogni singola directory o gerarchia di directories se non
# si vuole che la direttiva sopra sia considerata di default
#

#
# Definiamo delle regole di default per quanto riguarda il
# DocumentRoot
#
<Directory "/var/www/htdocs">

#
# La direttiva Options controlla le caratteristiche che possono
# essere abilitate in un certo contesto, le caratteristiche
# possono essere: "None", "All", o qualsiasi combinazione di
# "Indexes", "Includes", "FollowSymLinks", "ExecCGI", or
```

```
# "MultiViews".
#
# E' possibile trovare una trattazione dettagliata delle
# caratteristiche che possono essere abilitate con Options
# al link: http://httpd.apache.org/docs/mod/core.html#options
#
# "MultiViews" deve essere dichiarata esplicitamente,
# l'eventuale "Options All" da solo non e' infatti sufficiente
#
    Options Indexes FollowSymLinks MultiViews

#
# Qui sono indicate le direttive che i file .htaccess contenuti
# nelle directories possono scavalcare.
# Valori possibili sono: "All", o qualsiasi combinazione di
# "Options", "FileInfo", "AuthConfig" e "Limit"
#
# A questo link e' disponibile una trattazione dettagliata delle
# caratteristiche che possono essere abilitate con Options:
# http://httpd.apache.org/docs/mod/core.html#allowoverride
#
    AllowOverride AuthConfig

#
# Order: indica l'ordine con cui vengono valutate le direttive
# di Allow e Deny.
#
# Allow: Indica quali host possono accedere al server, nel nostro
# caso tutti. Eventualmente puo' comparire una lista di nomi di
# host separata da spazi
#
    Order allow,deny
    Allow from all
</Directory>

#
# UserDir: il nome della directory che ogni utente puo'
# utilizzare, a partire dalla suo directory personale, per
# pubblicare documenti via web.
# Se l'utente ad esempio si chiama mp e se la sua directory
# personale e' /home/mp/, i suoi documenti saranno pubblicabili
# inserendoli in /home/mp/public_html e saranno accessibili
# attraverso l'URL:
# http://host.tld/~mp
#
# Nel nostro caso la direttiva e' disponibile per il solo utente
```

```
# pm
#
<IfModule mod_userdir.c>
    UserDir public_html
    UserDir disabled *
    UserDir enabled pm
</IfModule>
#

# Controllo dell'accesso alle directories UserDir, ci sono
# direttive che abbiamo gia' incontrato
#
<Directory "/home/*/public_html">
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes
        Order allow,deny
        Allow from 192.168.1.2
</Directory>

#
# DirectoryIndex: nome dei files di default, separati da spazi,
# da caricare come indici delle directories.
#
<IfModule mod_dir.c>
    DirectoryIndex index.html index.php
</IfModule>

#
# AccessFileName: il nome del file da inserire nelle directory
# per definire l'accesso agli stessi
#
AccessFileName .htaccess

#
# Le linee successive evitano che i files .htaccess possano
# essere caricati dai browser, si tratta di espressioni
# regolari.
#
# Questo funzionera' anche per i files .htpasswd, ovvero il nome
# dei files che contengono le password di autenticazione
#
# Satisfy: e' la policy applicata (All o Any) se per una singola
# risorsa vengono indicati sia il controllo sugli accessi basato
# sull'host che l'autenticazione, dunque in caso di conflitto.
# Puo' essere usato any ad esempio per permettere l'accesso non
# autenticato alle richieste che provengono da macchine su
```

```
# una rete sicura.
#
# L'espressione regolare sta ad indicare tutti i file che
# iniziano per .ht
#
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>


# UseCanonicalName: se questa direttiva e' settata a On Apache
# utilizzera' il ServerName e la Port per formare un nome
# canonico, nel caso della generazione di URL autoreferenzianti.
# Se invece e' settata a Off Apache utilizzera' hostname:porta
# che fornira' il client, laddove sia possibile (da utilizzare
# quindi nel caso di hosting virtuale in cui appunto l'indirizzo
# autoreferenziante e' fornito dal client stesso per mezzo delle
# estensioni introdotte nel protocollo HTTP/1.1).
# Questa direttiva influenza anche le variabili CGI SERVER_NAME e
# SERVER_PORT negli scripts CGI
#
UseCanonicalName On

#
# Imposta il nome dei file di configurazione dei tipi MIME.
#
<IfModule mod_mime.c>
    TypesConfig /etc/apache/mime.types
</IfModule>

#
# DefaultType: tipo MIME di default che il server utilizzera'
# per i documenti che non ne specificheranno uno o che non sia
# riconducibile dall'estensione del file.
#
DefaultType text/plain

#
# MIMEMagicFile: indica il nome del file magic MIME ed abilita il
# modulo mod_mime_magic che permette al server di rilevare il
# contenuto MIME di un determinato file.
#
<IfModule mod_mime_magic.c>
    MIMEMagicFile /etc/apache/magic
```

```
</IfModule>

#
# HostnameLookups: abilita o disabilita le ricerche sul DNS
#
HostnameLookups Off

#
# ErrorLog: specifica il file dove vengono loggati gli errori
# Se non specificheremo per ogni host virtuale una propria
# direttiva ErrorLog, tutti gli errori saranno qui loggati.
#
# TransferLog: indica come loggare i trasferimenti avvenuti tra
# client e server
#
# I log vengono inviati tramite pipe a cronolog che effettua la
# rotazione dei log, senza usare la scomoda notazione dei file
# di rotatelogs, infatti cronolog suddivide i log in file e
# directories in base alla data formattata in formato "human
# readable"
#
TransferLog "|/usr/sbin/cronolog \
   /var/log/apache/%Y/%m/%d/access_log"
ErrorLog "|/usr/sbin/cronolog /var/log/apache/%Y/%m/%d/error_log"

# LogLevel: controlla il livello di loquacita' del logger.
# Possibile valori sono: debug, info, notice, warn, error,
# crit, alert, emerg, ovvero gli stessi livelli configurabili
# con syslogd
#
LogLevel warn

#
# Specifichiamo come deve essere il formato dei log
# (il carattere '\' permette di far proseguire una linea di
# configurazione nella riga successiva)
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
  \  \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# Formato e posizione file degli accessi (Common Logfile Format).
# Anche in questo caso se non definiremo file di accesso
# per gli host di accesso, i loro log finiranno qui
```

```
#
# Preferisco avere un logfile che contenga sia i referer sia
# gli agent dunque utilizzo la direttiva sotto
#
CustomLog /var/log/apache/access_log combined

#
# Per avere un logfile per i referer ed uno per gli agent
# *divisi* decommentare sotto e commentare sopra
#
#CustomLog /var/log/apache/referer_log referer
#CustomLog /var/log/apache/agent_log agent

#
# ServerSignature: genera una linea di footer che contiene
# l'indirizzo del server nei documenti generati dal server
# stesso.
# Settato su "EMail" include anche un link del tipo mailto: al
# ServerAdmin. Puo' anche assumere i seguenti valori On | Off
#
ServerSignature On


#
# Aliases: aggiungiamo qui tutti gli alias di cui abbiamo
# bisogno.
# Gli alias permettono di mappare URL a nomi di file.
# Il formato e' il seguente: Alias fakename realname
#
<IfModule mod_alias.c>

    #
    # Se includiamo un / terminale nel fakename il server
    # richiedera' che lo stesso sia presente nell'URL.
    # Dunque occorre TOGLIERE IL / in Alias dopo il nome
    # dell'alias (il fakename)in modo che non sia necessario
    # null'URL
    #
    Alias /icons/ "/var/www/icons/"

    <Directory "/var/www/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

```
# Questi Alias sono ASSOLUTAMENTE NECESSARI altrimenti con
# la direttiva VirtualDocumentRoot che c'e' in basso non
# funzionerebbero i vari siti come:
# http://linux2/mp

Alias /mp "/var/www/htdocs/mp/"
<Directory "/var/www/htdocs/mp">
    Options Indexes FollowSymlinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

Alias /phpmyadmin "/var/www/htdocs/phpmyadmin/"
<Directory "/var/www/htdocs/phpmyadmin">
    Options Indexes FollowSymlinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

Alias /webalizer "/var/www/htdocs/webalizer/"
<Directory "/var/www/htdocs/webalizer">
    Options Indexes FollowSymlinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

# ScriptAlias: definisce le directories che contengono
# script CGI.
# Il comportamento e' simile a quello della direttiva Alias
# quindi si applicano le stesse regole per gli slash
# applicate ad alias
#
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

```
</IfModule>
# Fine degli aliases

# Redirect ci permette di mappare URL su altri URL, nel caso ad
# esempio che un nostro URL sia cambiato.
# Il formato e' il seguente:
#
# Redirect [status] old-URI new-URL
#
# status puo' assumere diversi valori consultabili nel manuale
#

#
# Le direttive seguenti controllano l'indicizzazione dei file
# contenuti nelle directories (il modulo che permette
# l'indicizzazione dei file laddove non sia presente il file
# DirectoryIndex e' mod_autoindex)
#

<IfModule mod_autoindex.c>

    #
    # Abilita il FancyIndexing
    #
    IndexOptions FancyIndexing

    #
    # AddIcon* mappa le icone ai nome file nelle directory
    # in cui e' abilitato il FancyIndexed directories
    #
    AddIconByEncoding (CMP,/icons/compressed.gif) \
       x-compress x-gzip

    AddIconByType (TXT,/icons/text.gif) text/*
    AddIconByType (IMG,/icons/image2.gif) image/*
    AddIconByType (SND,/icons/sound2.gif) audio/*
    AddIconByType (VID,/icons/movie.gif) video/*

    AddIcon /icons/binary.gif .bin .exe
    AddIcon /icons/binhex.gif .hqx
    AddIcon /icons/tar.gif .tar
    AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
    AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
    AddIcon /icons/a.gif .ps .ai .eps
    AddIcon /icons/layout.gif .html .shtml .htm .pdf
```

```
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon: icona di default per i file che non ne hanno
# una configurata appositamente
#
DefaultIcon /icons/unknown.gif

#
# AddDescription: piccola descrizione del tipo file laddove
# sia abilitato il FancyIndexed'ing'
# directories.
#
AddDescription "GZIP compressed document" .gz
AddDescription "tar archive" .tar
AddDescription "GZIP compressed tar archive" .tgz
AddDescription "File di mp" .mp

#
# ReadmeName: il file che sara' aggiunto, nel
# caso sia presente, al termine della pagina web nelle
# indicizzazioni delle directories
#
# HeaderName: il file che sara' aggiunto, nel
# caso sia presente, all'inizio della pagina web nelle
# indicizzazioni delle directories
#
ReadmeName README
HeaderName HEADER

#
# IndexIgnore: l'insieme dei nome dei file che devono essere
# ignorati nelle indicizzazioni delle directories
```

```
    #
    IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

</IfModule>
# Fine delle direttive di indicizzazione

#
# Tipi di documento (tipi MIME)
#
<IfModule mod_mime.c>

    #
    # AddEncoding: mappa le estensioni al tipo di codifica
    #
    AddEncoding x-compress Z
    AddEncoding x-gzip gz tgz

    #
    # AddLanguage: permette di specificare la lingua di un
    # documento, permette poi al browser di inviare i giusti
    # documenti in base alla lingua (in alcuni casi)
    #
    # AddCharset: mappa le estensioni con i tipi di set di
    # caratteri
    #
    # Danish (da) - Dutch (nl) - English (en) - Estonian (ee)
    # French (fr) - German (de) - Greek-Modern (el)
    # Italian (it) - Korean (kr) - Norwegian (no) – Norwegian
    # Nynorsk
    # (nn) - Portugese (pt) - Luxembourgeois* (ltz)
    # Spanish (es) - Swedish (sv) - Catalan (ca) - Czech(cz)
    # Polish (pl) - Brazilian Portuguese (pt-br) - Japanese (ja)
    # Russian (ru)
    #
    AddLanguage da .dk
    AddLanguage nl .nl
    AddLanguage en .en
    AddLanguage et .ee
    AddLanguage fr .fr
    AddLanguage de .de
    AddLanguage el .el
    AddLanguage he .he
    AddCharset ISO-8859-8 .iso8859-8
    AddLanguage it .it
    AddLanguage ja .ja
    AddCharset ISO-2022-JP .jis
```

```
AddLanguage kr .kr
AddCharset ISO-2022-KR .iso-kr
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddCharset ISO-8859-2 .iso-pl
AddLanguage pt .pt
AddLanguage pt-br .pt-br
AddLanguage ltz .lu
AddLanguage ca .ca
AddLanguage es .es
AddLanguage sv .sv
AddLanguage cz .cz
AddLanguage ru .ru
AddLanguage zh-tw .tw
AddLanguage tw .tw
AddCharset Big5         .Big5    .big5
AddCharset WINDOWS-1251 .cp-1251
AddCharset CP866        .cp866
AddCharset ISO-8859-5   .iso-ru
AddCharset KOI8-R       .koi8-r
AddCharset UCS-2        .ucs2
AddCharset UCS-4        .ucs4
AddCharset UTF-8        .utf8

# LanguagePriority:permette di dare la precedenza tra le
# varie lingue.
#
<IfModule mod_negotiation.c>
    LanguagePriority it en fr de da nl et el ja kr \
                no pl pt pt-br ru ltz ca es sv tw
</IfModule>

#
# AddType: permette di mappare le estensioni ai tipi MIME
#
AddType application/x-tar .tgz

#
# AddHandler: permette di mapparre estensioni di file a un
# determinato gestore (handler)
# Se si vogliono usare i Server Side Includes o i CGI occorre
# decommentare la riga seguente
#
AddHandler cgi-script .cgi .pl
```

```
</IfModule>
# Fine tipi di documento (tipi MIME)

#

# Action: indica lo script da avviare se il tipo MIME o il
# gestore corrispono alla determinata risorsa richiesta
#
# Formato: Action media/type /cgi-script/location
# Formato: Action handler-name /cgi-script/location
#

#
# MetaDir: nome della sotto-directory che contiene i file delle
# metainformazioni
#
#MetaDir .web

#
# MetaSuffix: suffisso dei nomi dei file che contengono le
# metainformazioni
#
#MetaSuffix .meta

#
# Risposte agli errori personalizzabili (Apache style)
#
#    1) testo ASCII
#ErrorDocument 500 "The server made a boo boo.
#  attenzione alle singole virgolette (")
#
#    2) redirezioni locali verso altri URL che gestiscono
#       l'output da inviare all'utente
#ErrorDocument 404 /missing.html
#
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#
#
#    3) redirezioni esterne
#ErrorDocument 402 http://other-server.com/subscription_info.html

#
# Customize behaviour based on the browser
#
<IfModule mod_setenvif.c>
```

```
    #
    # Le seguenti direttive modificano il comportamento delle
    # risposte HTTP per eliminare alcuni errori dei browsers
    #
    BrowserMatch "Mozilla/2" nokeepalive
    BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 \
          force-response-1.0

    #
    # Le seguenti direttive disabilitano le risposte HTTP/1.1 a
    # determinati client, forzando quelle HTTP/1.0
    #
    BrowserMatch "RealPlayer 4\.0" force-response-1.0
    BrowserMatch "Java/1\.0" force-response-1.0
    BrowserMatch "JDK/1\.0" force-response-1.0

</IfModule>
# Fine delle direttive personalizzate per i client

#
# Abilita il "server status reports", raggiungibile
# all'URL http://servername/server-status
#
# Attne: *non* si deve creare nessuna directory server-status in
#        DocumentRoot
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 192.168.1.2
</Location>

#
# Abilita il "remote server configuration reports", raggiungibile
# all'URL http://servername/server-info (richiede mod_info.c
# caricato)
#
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .your-domain.com
</Location>

#
# Direttive del Proxy Server. Decommentare per abilitare il
```

```
# proxy server. Eventualmente si puo' associare una porta
# specifica al proxy invece della 80, grazie alla direttiva
# Container <VirtualHost>
#
<IfModule mod_proxy.c>
    ProxyRequests On

    <Directory proxy:*>
        Order deny,allow
        Deny from all
        Allow from .mydomain
    </Directory>

    #
    # Abilita/disabilita la gestione di HTTP/1.1 "Via:" headers.
    # ("Full" aggiunge la versione del server; "Block" rimuove
    # tutti i Via: headers uscenti)
    # Set to one of: Off | On | Full | Block
    #
    ProxyVia On

    #
    # Per abilitare e configurare la gestione della cache
    # locale del Proxy
    #
    CacheRoot "/var/cache/proxy"
    CacheSize 5
    CacheGcInterval 4
    CacheMaxExpire 24
    CacheLastModifiedFactor 0.1
    CacheDefaultExpire 1
    NoCache a-domain.com another-domain.edu joes.garage-sale.com

</IfModule>

# Fine dele direttive proxy


### Sezione 3: Virtual Hosts
#
# VirtualHost: per poter gestire piu' siti web sullo stesso
# server.
#
# Leggete la seguente documentazione:
# <URL:http://www.apache.org/docs/vhosts/>
#
```

```
#
# Per utilizzare l'hosting basato su nome, il piu' utilizzato,
# che sfrutta un header speciale di richiesta introdotto con
# HTTP/1.1
#
#NameVirtualHost 192.168.1.2

#
# Un esempio completo di utilizzo della direttiva VirtualHost,
# host.example.com e' l'host virtuale,
#
#
#<VirtualHost FQDN-server-locale>
#    ServerAdmin webmaster@host.example.com
#    DocumentRoot /www/docs/host.example.com
#    ServerName host.example.com
#    ErrorLog logs/host.example.com-error_log
#    CustomLog logs/host.example.com-access_log common
#</VirtualHost>

#
# Un altro esempio, IP locale del server, host.dom.com e' l'host
# virtuale
#
#<VirtualHost 192.168.1.2>
#    ServerAdmin webmaster@host.dom.com
#    DocumentRoot /www/docs/host.dom.com
#    ServerName host.dom.com
#    ErrorLog logs/host.dom.com-error_log
#    CustomLog logs/host.dom.com-access_log common
#</VirtualHost>

#
# Virtual hosting dinamico
#
# accesso diretto alle directories sotto /var/www/htdocs
# Esempio:  w.io.dom accede a index.html nella directory
# /var/www/htdocs/w.io.dom
# ATT.NE l'host deve essere registrato su DNS o su /etc/hosts

UseCanonicalName off
VirtualDocumentRoot /var/www/htdocs/%0


# Tutti i moduli esterni sono di default disabilitati, a meno
```

```
# che non si introduca una direttiva nel file di
#configurazione, che fa si che il modulo sia caricato.
# E' possibile anche caricare esplicitamente i moduli in questa
# sottosezione. Nell'esempio vengono letti tramite la
# direttiva include i file che contengono al loro interno le
# direttive per caricare i rispettivi moduli e per configurarli
# ( ad esempio AddModule e AddType per PHP)
#
# ==> mod_php configuration settings <==
#
#                -- Note specifiche per Slackware --
# PACCHETTO RICHIESTO:  openssl-solibs (A series) e/o openssl
#                       (N series), mysql (AP series), gmp
#                       (L series) e apache (N series)
#
Include /etc/apache/mod_php.conf

# ==> mod_ssl configuration settings <==
#
#                -- Note specifiche per Slackware --
# PACCHETTO RICHIESTO:  apache (N series) e openssl (N series)
#
Include /etc/apache/mod_ssl.conf
```

10. Amministrazione

Ora che il server web Apache e' reso operativo a seguito dell'installazione e della opportuna configurazione, e' necessaria una periodica manutenzione e gestione da parte dell'amministratore di sistema.

Le attivita' periodiche da effettuare sono in linea di massima le seguenti:

1. controllo e rotazione dei log file
2. eventuale modifica della configurazione
3. eventuale aggiornamento del software

1) Per la visualizzazione dei file di log in maniera grafica, piu' "user friendly" e' possibile utilizzare **webalizer**, mentre per la rotazione degli stessi e' consigliabile l'utilizzo di **cronolog**. Il controllo di Apache avviene utilizzando i comandi **httpd** e **apachectl**, attraverso l'uso di strumenti grafici di terze parti o built-in.

2) La modifica della configurazione puo' essere effettuata "a mano" editando direttamente i file o utilizzando strumenti **GUI** (Graphical User Interface) di configurazione e amministrazione come **webmin**.

3) L'upgrade della versione di Apache dipende dalla distribuzione e dal fatto che si sia optato per la compilazione da sorgenti o per l'installazione di un pacchetto binario.

Controllo manuale

Il comando **httpd** fornisce molte informazioni utili per il controllo di Apache. Procediamo come al solito con un insieme di esempi.

Mostra la versione di Apache
 # httpd -v

Mostra le opzioni di compilazione
 # httpd -V

Elenca i moduli compilati staticamente
 # httpd -l

Elenca le direttive disponibili
 # httpd -L

Elenca gli host virtuali
 # httpd -S

Controlla la sintassi del file di configurazione
 # httpd -t

Un frontend per **httpd** e' lo shell script **apachectl**, il suo utilizzo e' molto intuitivo, il comando seguente vi permette di avere una lista delle opzioni:
 # apachectl

Controllo mediante interfaccia grafica (GUI)

Esistono diversi strumenti per la configurazione di Apache per mezzo di interfaccia grafica, il loro utilizzo e' banale (naturalmente e' necessario conoscere le direttive di Apache):

1. **comanche**
2. **webmin**

Controllo mediante strumenti built-in

Nel file di configurazione precedentemente analizzato sono presenti dei moduli che forniscono informazioni dettagliate sullo stato del web server; si tratta dei moduli **mod_status** e **mod_info**.
Mediante le seguenti direttive vengono rese disponibili informazioni sullo stato del web server:

```
.......
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 192.168.1.2
</Location>
.......
```

Le informazioni sono visibili all'URL: http://192.168.1.2/server-status
Al posto dell'indirizzo IP potete naturalmente inserire il vostro FQDN.

Le seguenti direttive invece vi permettono di avere informazioni dettagliate sulla configurazione del web server:

```
......
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .your-domain.com
</Location>
.......
```

Le informazioni sono visibili all'URL: http://192.168.1.2/server-info
anche qui potete inserire il vostro FQDN in luogo dell'indirizzo IP.

11. Sicurezza: Autenticazione

Autenticazione utenti: principi

Nel file di configurazione della pagina precedente abbiamo visto come limitare l'accesso a determinati host o insiemi di host (sottoreti) mediante le direttive **Allow** e **Deny**; vediamo ore come fare per effettuare l'autenticazione degli utenti per restringere l'accesso a determinati documenti web.

E' possibile utilizzare l'autenticazione HTTP descritta nello RFC 2617 "HTTP Authentication: Basic and Digest Access Authentication". Qui tratteremo la sola autenticazione "Basic".

Dal lato client l'autenticazione avviene introducendo nome utente e password in una finestra che viene aperta in automatico dal browser non appena si cerca di accedere alla risorsa protetta.

Il server web infatti in questo caso inviera' il seguente messaggio:

HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="StringaContenutaInAuthname"
Server: linux2
Date: 01-01-04
Content-type: text/html
Content-length ...

che interpretato dal client causera' l'apertura automatica della finestra nel browser, con titolo specificato in **realm.**

Se il nome utente risulta non essere valido, o se la password e' errata, l'accesso alla risorsa viene negato ed il server invia il messaggio di errore 401 (accesso non autorizzato).

Configurare Apache per ottenere l'autenticazione di tipo Basic.

Controlliamo che il modulo **mod_auth** sia specificato con la direttiva LoadModule nel file httpd.conf .

Prima dobbiamo creare gli utenti ai quali permettere l'accesso alle risorse protette ed assegnargli delle password: questo e' possibile per mezzo del comando **htpasswd**:

 # htpasswd -bc /var/www/htdocs/.passwd UTENTE PASSWORD

In questo modo abbiamo creato l'utente UTENTE, gli abbiamo assegnato la password PASSWORD ed abbiamo memorizzato il tutto nel file `.htpasswd` contenuto nella DocumentRoot.

Ora facciamo in modo che venga ristretto l'accesso ad una determinata risorsa web, ovvero ad una intera directory contenuta nella DocumentRoot, inserendo in essa un file di nome `.htaccess` con il seguente contenuto:

```
AuthName "L O G I N"
AuthUserFile /var/www/htdocs/.htpasswd
AuthType Basic
require user UTENTE
```

AuthName sara' il titolo che comparira' sulla finestra del browser (quella specificata in BasicRealm dal server web), **AuthUserFile** e' il nome del file che contiene nome utente e password criptata, file creato precedentemente con **htpasswd**, **AuthType** specifica il tipo di autenticazione (Basic nel nostro caso), **require user** permette di specificare a quali utenti permettere l'autenticazione.

Esistono altri tipi di autenticazione, ad esempio attraverso il database **Mysql**, per il quale esiste il modulo **mod_auth_mysql**: si tratta di una soluzione piu' efficiente nel caso di server ad elevato traffico, oltre che di una soluzione piu' sicura rispetto all'uso di un file di testo.

Autenticazione utenti (script lato server)

Per l'autenticazione degli utenti in realta' i moduli di Apache sono sempre meno utilizzati, grazie alle sempre maggiori funzionalita' offerte dai linguaggi di scripting server side come PHP.

Vediamo in proposito il seguente esempio, un file HTML con linguaggio PHP embedded:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
 <title>:: Accesso Amministratore ::</title>
</head>

<body>
```

```
<br>
 <h2>
   Accesso amministratore del sito
 </h2>
<br>
<br>

<hr size="5" width="90%">

<p>

<?php

$user = "NomeUtente";
$pwd = "PasswordUtente";

if (isset($_SERVER['PHP_AUTH_USER'])) {
    $PHP_AUTH_USER = $_SERVER['PHP_AUTH_USER'];
}
if (isset($_SERVER['PHP_AUTH_PW'])) {
    $PHP_AUTH_PW = $_SERVER['PHP_AUTH_PW'];
}

function autentica(){
    $realm="Titolo finestra: Amministrazione";

    header("WWW-Authenticate: Basic realm='".$realm."'");
    header("HTTP/1.0 401 Unauthorized");

     echo "\n<font color=\"red\">\n";
     echo "\n<b>Non potete accedere a questa pagina</b>\n";
     echo "\n</font>\n";

    exit();
}


if( !isset($PHP_AUTH_USER) && !isset($PHP_AUTH_PW) ) {
    autentica();
}
else {
    if( $PHP_AUTH_USER==$user && $PHP_AUTH_PW==$pwd ) {
          echo "\nAccesso consentito.\n";
    }
    else{
        // richiama la funzione di identificazione
```

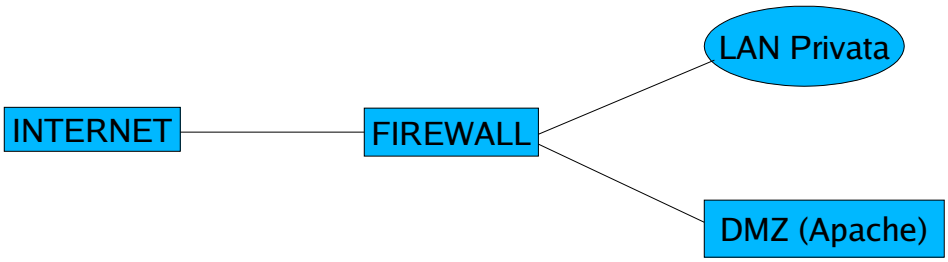
```
        autentica();
    }
}

?>
```

Salvare il file con estensione `.php` nella DocumentRoot di Apache ed aprirlo con un browser: si avra' lo stesso effettodi cio' che si era ottenuto con il modulo di Apache.

Con la tecnica di autenticazione appena esposta (Basic) il nome utente e la password viaggiano nelle Rete in chiaro, chiunque si trovi fisicamente collegato tra client e web server puo' potenzialmente **sniffare** i pacchetti e dunque entrare in possesso di informazioni riservate. Per i motivi appena esposti e' stata introdotta la **Digest Authentication** in HTTP 1.1 (**rfc** 2069), la quale permette di cifrare le informazioni sensibili grazie all'algoritmo **MD5**.

12. Sicurezza:
architettura tipica di un firewall in presenza di un web server

Il web server deve essere posto in modo tale che gli utenti remoti possano accedere ad esso, quindi e' potenzialmente esposto ad attacchi esterni.
Tipicamente l'architettura del firewall deve prevedere una **DMZ** (De-Militarized Zone) dove poter porre il web server:



Le regole del filtraggio dei pacchetti, realizzate grazie al pacchetto **iptables**, dovranno fare in modo che i pacchetti destinati alla porta TCP/80 siano indirizzati alla DMZ dove si trova Apache, andando a modificare le regole della tabella **NAT.**

Ad esempio se la DMZ ha indirizzo IP 192.168.1.2 la regola sara' del tipo:

*#iptables -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0 -j DNAT *
--to-destination 192.168.1.2:80

questo fara' si che tutto il traffico in ingresso all'interfaccia eth0, attraverso **socket** TCP, destinati alla porta 80, siano rediretti alla DMZ con indirizzo 192.168.1.2 porta 80.

Marco Pagnanini
http://www.marcopagnanini.it