APACHE

http://weblab.ing.unimo.it/retilab.shtml

http://weblab.ing.unimo.it/retilab/httpd-2.2.11.tar.bz2

Download dei sorgenti da httpd.apache.org

- i sorgenti sono disponibili anche sul sito del corso

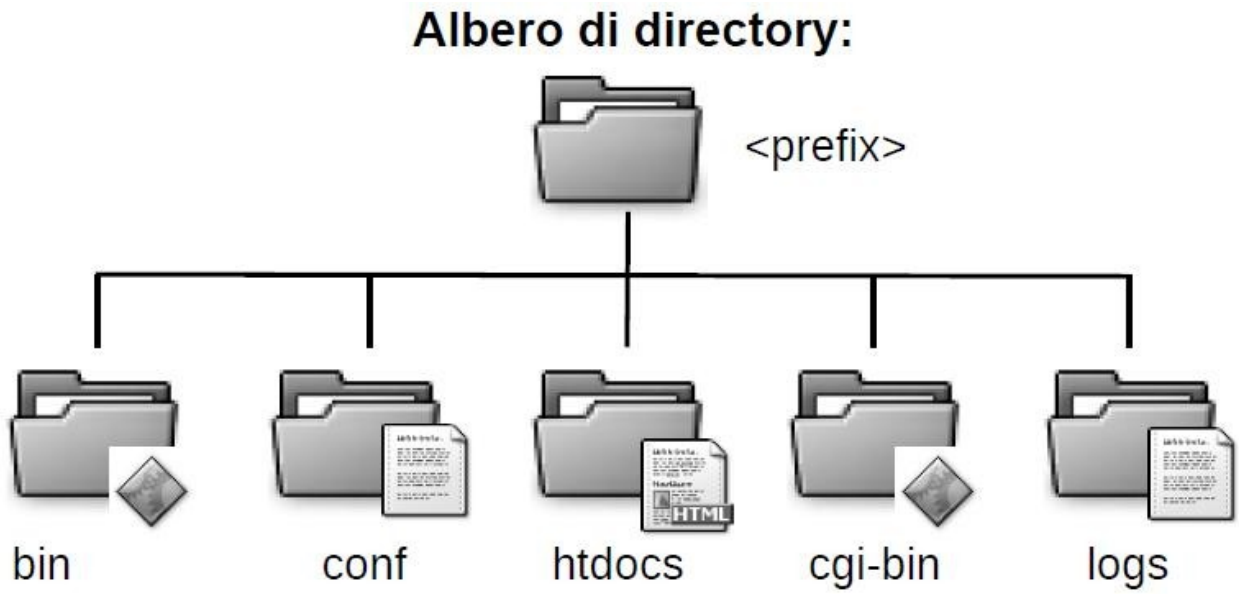
Noi facciamo riferimento alla versione 2.2 di apache

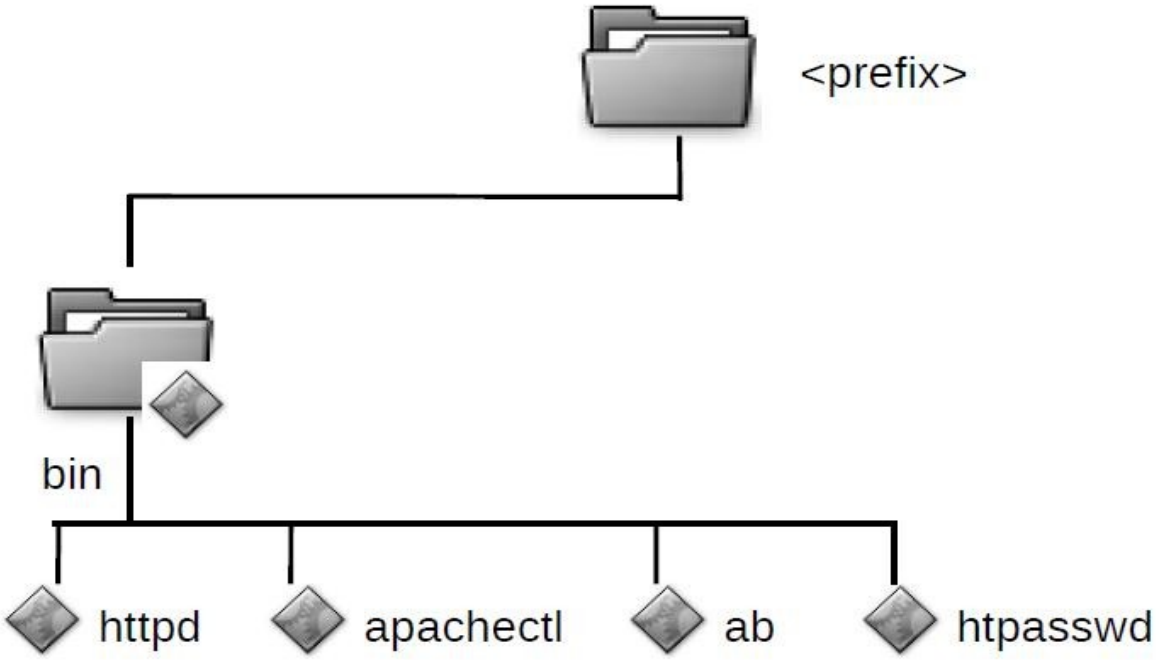
```
$ tar -xzvf httpd-<versione>.tar.gz
$ cd httpd-<versione>
$ ./configure --prefix=</destination/path>
$ make
$ make install
```

Opzioni da passare allo script configure

```
--prefix=<prefix>
--with-mpm={beos|worker|prefork|mpm_os2}
--enable-modules={...}
```

- **Prefix è la directory dove verrà installato Apache**
- **MPM indica quale sistema MPM dovrà essere usato per gestire le richieste concorrenti**
- **Enable modules indica quali moduli dovranno essere abilitati** (il default contiene i moduli più comunemente utilizati)



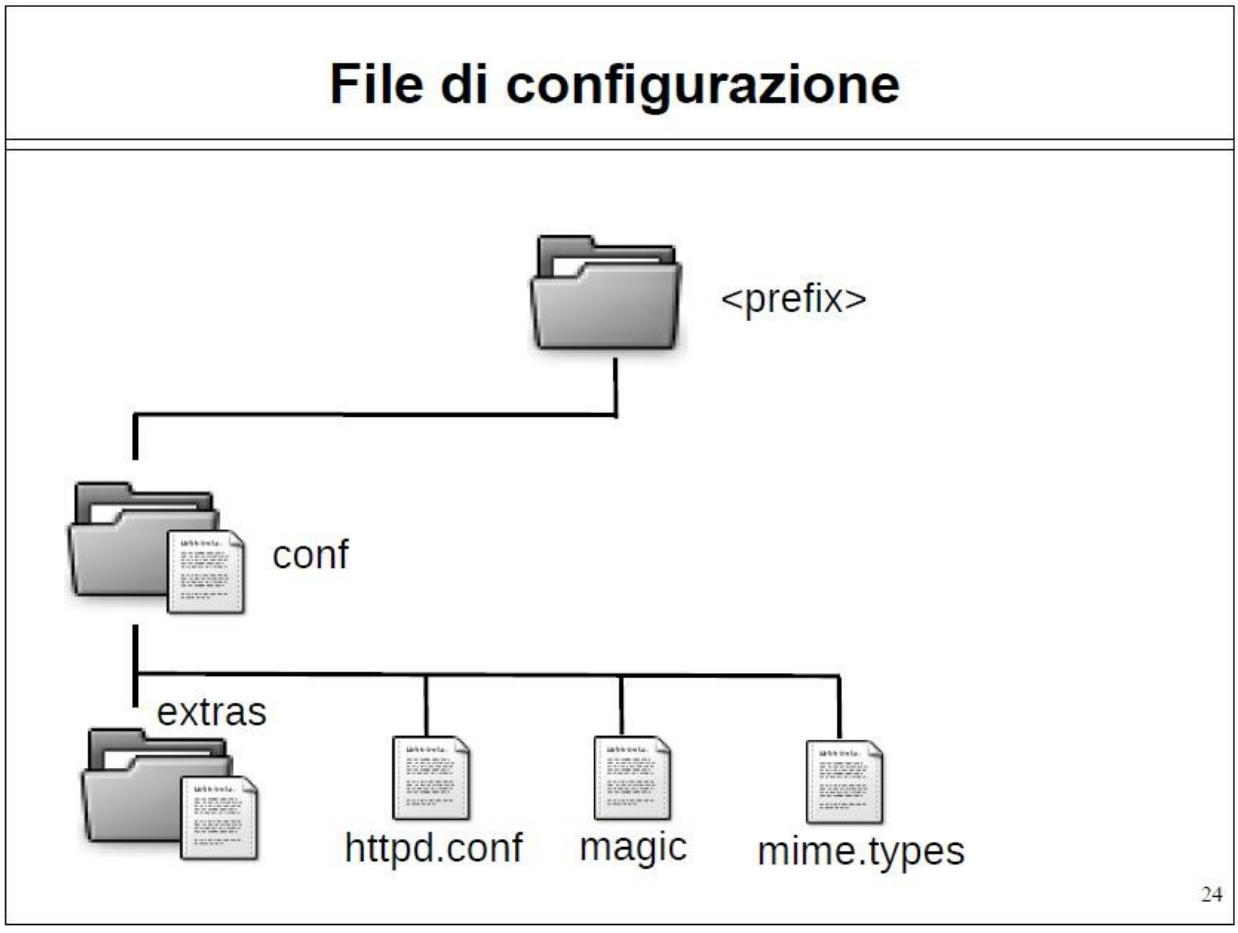


apachectl: script per lanciare, fermare e controllare il demone

- apachectl start
- apachectl stop
- apachectl restart

htpasswd: serve per generare e modificare file di autenticazione di apache

- htpasswd [-c] <file.passwd> <username>



File conf/httpd.conf
2 sezioni:

- impostrazioni globali
- impostazioni del server di default

Impostazioni globali

ServerRoot

- top level directory per altri file di configurazione

Listen

- porta su cui ascolta il server: nel nostro caso dovremo usare 8080

LoadModule

- Carica moduli esterni che estendono le funzioni di Apache (e.g., php)

User, Group

- Utente non privilegiato usato da Apache
- Per motivi di sicurezza si abbandonano il prima possibile i privilegi da *superutente*

MPM prefork

Processo principale
Processi aggiuntivi per servire le richieste
Parametri

- StartServers (numero di processi da far partire)
- MinspareServers, MaxSpareServers (autoadatta il numero di processi al carico, *ciclo di isteresi*)
- MaxClients (massimo parallelismo: evita il thrashing)
- MaxRequestPerChild (*rejuvination*: dopo un po' fa ripartire il processo per evitare il richio di memory leak)

MPM worker

Un processo principale
Processi ausiliari con thread multipli
Parametri

- ThreadsPerChild
- ServerLimit, ThreadLimit (massimo parallelismo: evita il thrashing)
- StartServers,MinspareServers, MaxSpareServers, (come modello prefork)

Verificare l'apertura della porta:

```
$ netstat -ntlp | grep 8080
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp6       0      0 :::8080       :::*    LISTEN 28526/httpd
```

Configurazione di log

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "logs/access_log" common
```

Significato dei campi:

- **%h** remote host
- **%l** remote log name (ottenuto mediante identd)
- **%u** remote user
- **%t** time of request
- **\"%r\"** prima linea della richiesta
- **%>s** stato della richiesta (response code)
- **%b** domensione della risposta in byte
- **\"%{Referer}i\"** header .referer. nella richiesta
- **\"%{User-Agent}i\"** header .user agent. nella richiesta

Esempio

Realizzare un meccanismo di autenticazione e controllo di accesso basato su username e password. Accessi alla directory “secret”devono essere consentiti solo all'utente “riccardo” autenticato mediante password.

Access control lists con password
AllowOverride (nel file di configurazione)

- AuthConfig (posso ridefinire le modalità di accesso)

Creazione file .htpasswd

- htpasswd(2) [-c] <file> <user>

Nel file .htaccess

- AuthType Basic (autenticazione con password)
- AuthName (.realm. di autenticazione)
- AuthUserFile, AuthGroupFile (dove si trovano informazioni per l'autenticazione)
- Require {valid-user | user <user> | group <group>}
  (citerio di autenticazione)

- **in httpd.conf**

AllowOverride Authconfig

- **file secret/.htaccess**

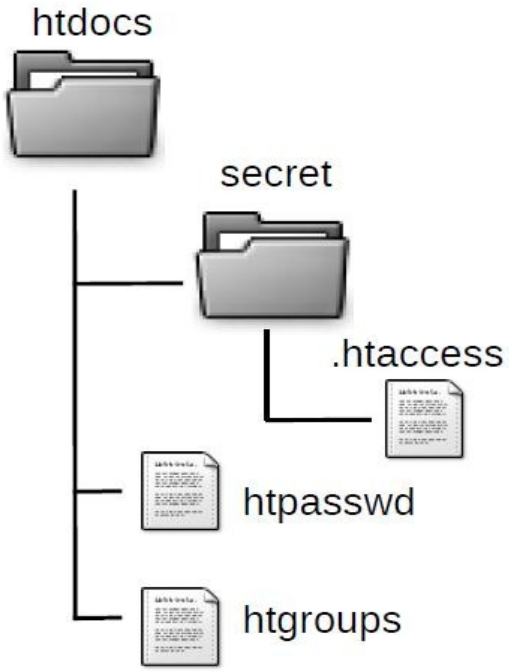
AuthType Basic
AuthName "Test"
AuthUserFile /<prefix>/htdocs/htpasswd
AuthGroupFile /<prefix>/htdocs/htgroups
Require group secretGroup

- **file htgroups**

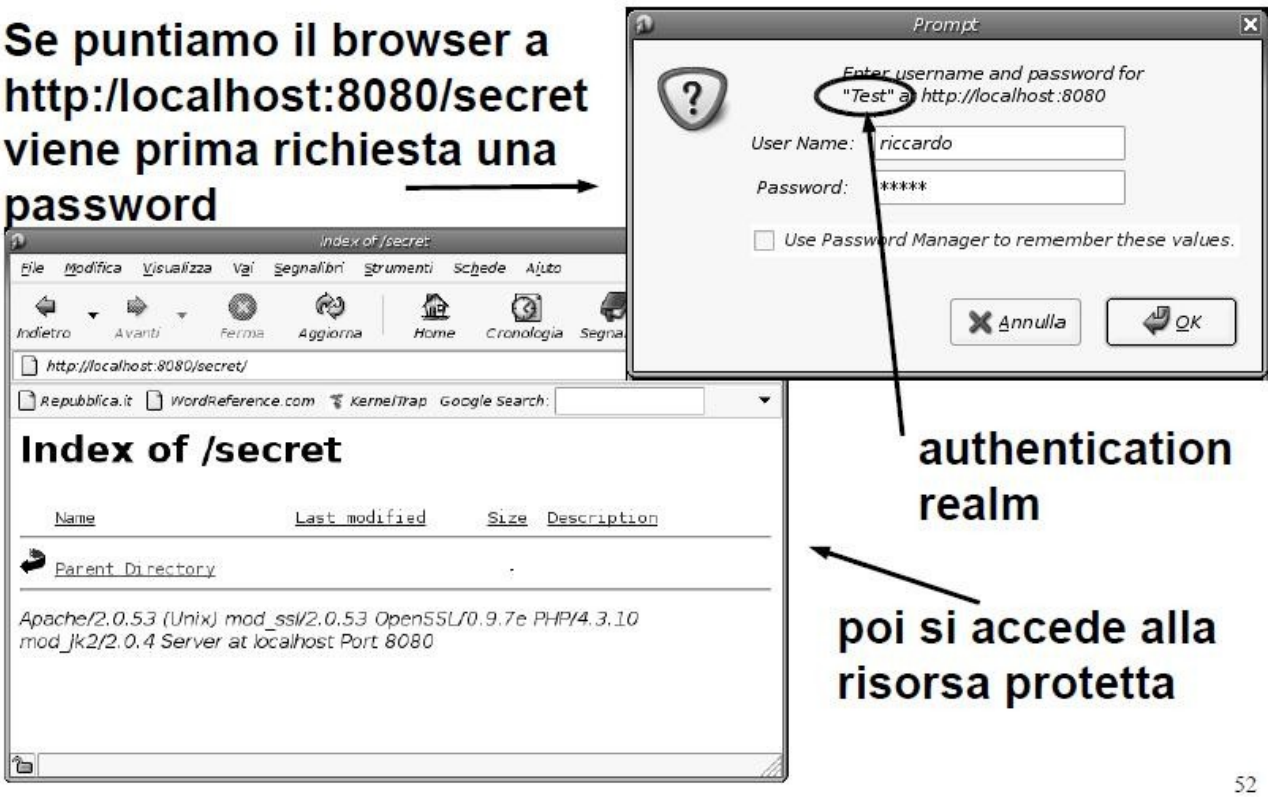
secretGroup: riccardo

- **file htpasswd**

riccardo:hLmwBbd3GiEsA



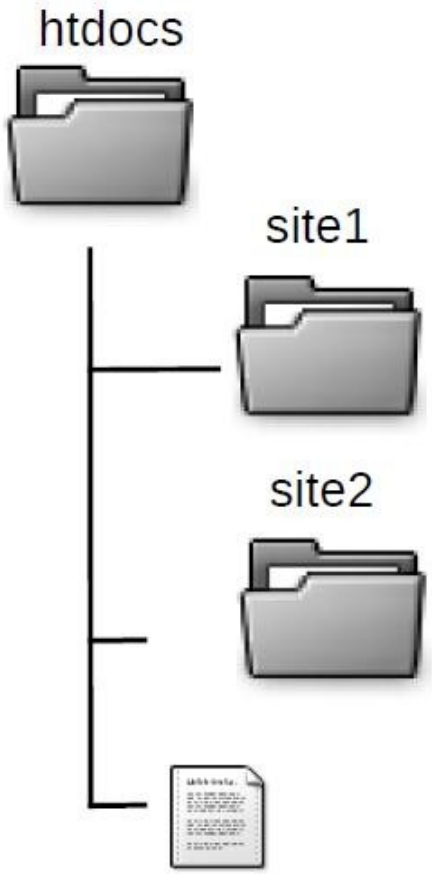
Se puntiamo il browser a http:/localhost:8080/secret viene prima richiesta una password



authentication realm

poi si accede alla risorsa protetta

- Realizzare un meccanismo di virtual hosting con 2 siti
- **w**ww.site1.com usa come document root la directory “site1”
- www.site2.com usa come document root la directory “site2”
- Il sistema usa name-baed virtual hosting




```
NameVirtualHost *:8080
<VirtualHost *:8080>
   ServerName localhost
</VirtualHost>
<VirtualHost *:8080>
   ServerAdmin webmaster@www.site1.com
   DocumentRoot htdocs/site1
   ServerName www.site1.com
</VirtualHost>
<VirtualHost *:8080>
   ServerAdmin webmaster@www.site2.com
   DocumentRoot htdocs/site2
   ServerName www.site2.com
   ErrorLog logs/site2_error.log
   CustomLog logs/site2_access.log combined
</VirtualHost>
```

Verifica del funzionamento
Con nc (o telnet) ci interfacciamo direttamente al protcollo HTTP

- Richiediamo prima www.site1.com
- Poi www.site2.com

Formato di una richiesta HTTP

GET <PATH> HTTP/1.1
Host: <host>

Nel nostro caso:

<PATH>=/

<host>=www.site1.com

oppure <host>=www.site2.com

Risultato se gli host funzionano correttamente:

$ nc localhost 8080
GET / HTTP/1.1
Host: www.site1.com

HTTP/1.1 200 OK
Date: Thu, 05 May 2005 20:45:23 GMT
Server: Apache/2.0.53 (Unix) mod_ssl/2.0.53 OpenSSL/0.9.7e PHP/4.3.10 mod_jk2/2.0.4
Last-Modified: Sat, 30 Apr 2005 10:32:39 GMT
ETag: "394946-61-e8efffc0"
Accept-Ranges: bytes
Content-Length: 97
Content-Type: text/html

<html>
<head><title>Site1 Home Page</title></head>
<body><h1>Site 1 content</h1></body>
</html>